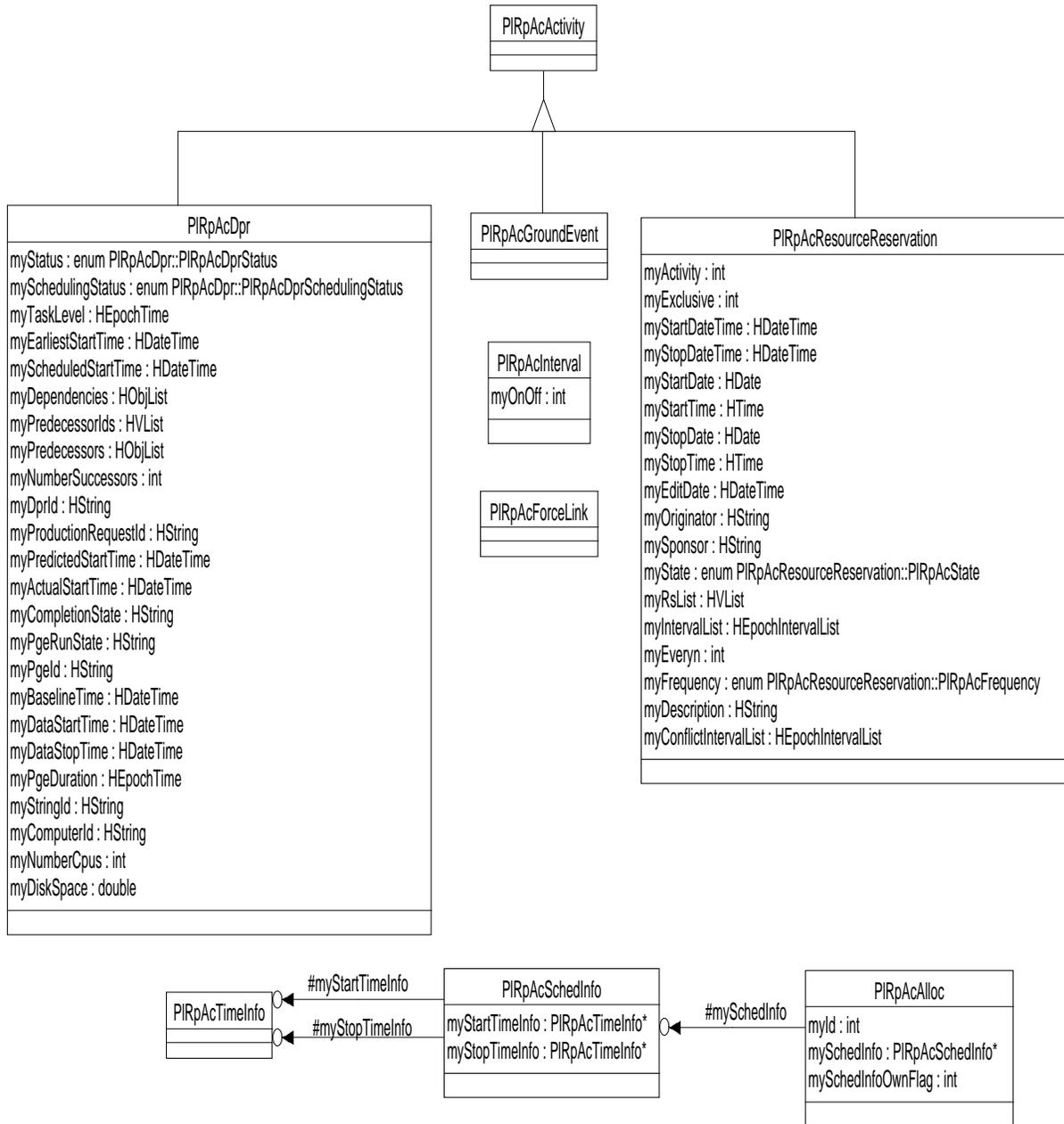


6.3.15 Resource_Planning_Ac Class Category

6.3.15.1 Overview



6.3.15.2 Resource_Planning_Ac Classes

6.3.15.3 PIRpAcActivity Class

Overview:

class PIRpAcActivity (-)

An instance of a PIRpAcActivity represents a request to change the configuration of a resource.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PIRpAcActivity();
```

Default constructor

```
PIRpAcActivity(const PIRpAcActivity& orig);
```

Copy constructor

```
virtual ~PIRpAcActivity();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpAcActivity );
```

class PIRpAcActivity (-)

An instance of a PIRpAcActivity represents a request to change the configuration of a resource.

- get

```
virtual int get(istream& istr);
```

Get myself onto a stream

- operator =

```
PIRpAcActivity& operator =(const PIRpAcActivity& orig);
```

Assignment operator

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Get/put myself onto an XDR stream

6.3.15.3.1 PIRpAcAlloc Class

Overview:

An instance of a PIRpAcAlloc is an allocation containing information as to when the allocation was created, and any parameters that have been modified from the default parameters.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myId: int
```

My id

```
mySchedInfoOwnFlag: int
```

Whether I own my scheduling information

```
mySchedInfo: PIRpAcSchedInfo*
```

My scheduling information

Constructors and Destructor:

```
PIRpAcAlloc();
```

Default constructor

```
PIRpAcAlloc(const PIRpAcAlloc& orig);
```

Copy constructor

```
virtual ~PIRpAcAlloc();
```

Destructor

Operations:

- cleanup

```
virtual void cleanup();
```

Clean up any things that are left over.

- copyOver

```
int copyOver(const PIRpAcAlloc& orig);
```

Clone and own the data in the source

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpAcAlloc );
```

An instance of a PIRpAcAlloc is an allocation containing information as to when the allocation was created, and any parameters that have been modified from the default parameters.

- get

```
virtual int get(istream& istr);
```

Get myself from a stream

- id

```
virtual int id(int id);
```

Set my id

- id

```
virtual int id() const;
```

Get my id

- operator =

```
PIRpAcAlloc& operator =(const PIRpAcAlloc& orig);
```

Assignment operator

- ownData

```
virtual int ownData();
```

Ask me to own my data.

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- relData

```
virtual int relData();
```

Ask me to release my data.

- schedInfo

```
virtual PIRpAcSchedInfo* schedInfo();
```

Get my scheduling info, I relinquish ownership on the get the caller becomes owner

- schedInfo

```
virtual int schedInfo(PIRpAcSchedInfo* info);
```

Set my scheduling info, I assume ownership on the set

- xdr

```
virtual int xdr(XDR* xdrs);
```

Get/put myself onto an xdr stream

6.3.15.3.2 PIRpAcDpr Class

Overview:

Instances of PIRpAcDpr are activities which represent Data Processing Requests.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpAcActivity

Attributes:

myActualStartTime: HDateTime

Actual start time (Persistent)

myBaselineTime: HDateTime

Plan baseline time (Persistent)

myCompletionState: HString

Completion state (Persistent)

myComputerId: HString

Computer ID (Persistent). This will be used if a string ID is not provided.

myDataStartTime: HDateTime

Data start time (Persistent)

myDataStopTime: HDateTime

Data stop time (Persistent)

myDependencies: HObjList

Data dependency granule IDs

myDiskSpace: double

Disk Space Required (Persistent)

myDprId: HString

The following members are stored in the database. (key identifier (Persistent))

myEarliestStartTime: HDateTime

Earliest start time. This should be calculated from the arrival times of external data.

myNumberCpus: int

Number of CPUs (Persistent)

myNumbersuccessors: int

Number of successor DPRs.

myPgeDuration: HEpochTime

PGE duration (Persistent)

myPgeId: HString

PGE id (Persistent)

myPgeRunState: HString

PGE run state (Persistent)

myPredecessorIds: HVList

Predecessor DPR activity IDs. The list is contains unique IDs.

myPredecessors: HObjList

Predecessor DPR activities. The list should contains unique pointers.

myPredictedStartTime: HDateTime

Predicted start date/time. This should be set when the plan is rolled over to processing.
(Persistent)

myProductionRequestId: HString

Parent production request identifier (Persistent)

myScheduledStartTime: HDateTime

Earliest scheduled time. This should be set by the scheduling algorithm for the currently selected plan.

mySchedulingStatus: enum PlRpAcDpr::PlRpAcDprSchedulingStatus

My scheduling status

myStatus: enum PlRpAcDpr::PlRpAcDprStatus

My overall status

myStringId: HString

String ID (Persistent) If this is not supplied, we will use the computer ID.

myTaskLevel: HEpochTime

My task level: The level of a node in a task graph is defined as the length in time of the longest path from the node to the exit node.

Constructors and Destructor:

PlRpAcDpr();

Default constructor

PlRpAcDpr(const PlRpAcDpr& orig);

Copy constructor

virtual ~PlRpAcDpr();

Destructor

Operations:

- actualStartTime

virtual const HDateTime& actualStartTime() const;

Get my actual start time

- actualStartTime

virtual int actualStartTime(const HDateTime& aActualStart);

Set my actual start time

- addDependency

virtual int addDependency(const HString& granuleId);

Add a granule ID on which I depend to list of granules.

- addPredecessor

virtual int addPredecessor(const HString& predecessorDprName);

Add DPR to list of predecessors by name.

- addPredecessor

```
virtual int addPredecessor(PIRpAcDpr* predecessorDpr);
```

Add DPR to list of predecessors by object.

- adjustEarliestStartTime

```
virtual int adjustEarliestStartTime(const HDateTime&  
aStartTime);
```

Adjust my earliest start time. Only make an adjustment if the passed in start time is later than the current earliest start time.

- adjustScheduledStartTime

```
virtual int adjustScheduledStartTime(const HDateTime&  
aStartTime);
```

Adjust my scheduled start time. Only make an adjustment if the passed in start time is later than the current scheduled start time.

- baselineTime

```
virtual int baselineTime(const HDateTime& aBaselineTime);
```

Set my baseline time

- baselineTime

```
virtual const HDateTime& baselineTime() const;
```

Get my baseline time

- cleanup

```
virtual void cleanup();
```

Delete member data.

- compare

```
virtual int compare(const RActivity& ract) const;
```

Compare RActivity members.

- compare

```
virtual int compare(const HObject& obj) const;
```

Compare HObject members.

- compare

```
virtual int compare(const PIRpAcDpr& dprAct) const;
```

Compare PIRpAcDpr members.

- completionState

```
virtual const HString& completionState() const;
```

 Get my completion state
- completionState

```
virtual int completionState(const char* aCompletionState);
```

 Set my completion state
- computerId

```
virtual int computerId(const char* aComputerId);
```

 Set my Computer ID
- computerId

```
virtual const HString& computerId() const;
```

 Get my Computer ID
- copyOver

```
virtual int copyOver(const RSimpleAct& act);
```

 Copy internals of source activity.
- copyOver

```
virtual int copyOver(const PIRpAcDpr& dpr);
```

 Copy internals of source dpr.
- dataStartTime

```
virtual const HDateTime& dataStartTime() const;
```

 Get my data start time
- dataStartTime

```
virtual int dataStartTime(const HDateTime& aDataStart);
```

 Set my data start time
- dataStopTime

```
virtual const HDateTime& dataStopTime() const;
```

 Get my data stop time
- dataStopTime

```
virtual int dataStopTime(const HDateTime& aDataStop);
```

 Set my data stop time

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpAcDpr );
```

Instances of PIRpAcDpr are activities which represent Data Processing Requests.

- dependencies

```
virtual HObjIter* dependencies();
```

Get an iterator over my list of input granules.

- diskSpace

```
virtual double diskSpace() const;
```

Get my disk space requirement

- diskSpace

```
virtual int diskSpace(double amount);
```

Set my disk space requirement

- dprId

```
virtual int dprId(const char* aDprId);
```

Set my DPR identifier

- dprId

```
virtual const HString& dprId() const;
```

Get my DPR identifier

- earliestStartTime

```
virtual int earliestStartTime(const HDateTime& aStartTime);
```

Set my earliest start time. This should be calculated from the predicted arrival of external data.

- earliestStartTime

```
virtual const HDateTime& earliestStartTime() const;
```

Get my earliest start time.

- get

```
virtual int get(istream& istr);
```

Get myself onto a stream

- numberCpus

```
virtual int numberCpus() const;
```

Get my required number of CPUs

- numberCpus

```
virtual int numberCpus(int number);
```

Set my required number of CPUs
- numberSuccessors

```
virtual int numberSuccessors() const;
```

Get my number of successor DPRs
- numberSuccessors

```
virtual int numberSuccessors(int nSuccessors);
```

Set my number of successor DPRs
- operator =

```
PlRpAcDpr& operator =(const PlRpAcDpr& orig);
```

Assignment operator
- pgeDuration

```
virtual int pgeDuration(double duration);
```

Set my PGE duration
- pgeDuration

```
virtual double pgeDuration() const;
```

Get my PGE duration
- pgeId

```
virtual int pgeId(const char* aPgeId);
```

Set my PGE identifier
- pgeId

```
virtual const HString& pgeId() const;
```

Get my PGE identifier
- pgeRunState

```
virtual const HString& pgeRunState() const;
```

Get my PGE run state
- pgeRunState

```
virtual int pgeRunState(const char* aPgeRunState);
```

Set my PGE run state

- predecessors

```
virtual HObjIter* predecessors();
```

Get an iterator over my list of predecessor DPR activities.

- predictedStartTime

```
virtual int predictedStartTime(const HDateTime& aPredictedStartTime);
```

Set my predicted start time. This should be set when the plan is rolled over to processing.

- predictedStartTime

```
virtual const HDateTime& predictedStartTime() const;
```

Get my predicted start time.

- productionRequestId

```
virtual int productionRequestId(const char* aPrId);
```

Set my parent production request identifier

- productionRequestId

```
virtual const HString& productionRequestId() const;
```

Get my parent production request identifier

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- resolvePredecessorIds

```
virtual void resolvePredecessorIds();
```

Try to resolve any predecessor DPR ids to DPRs from the activity pool.

- scheduledStartTime

```
virtual int scheduledStartTime(const HDateTime& aStartTime);
```

Set my scheduled start time. This attribute is a place holder used by the scheduling algorithm for the currently selected plan.

- scheduledStartTime

```
virtual const HDateTime& scheduledStartTime() const;
```

Get my scheduled start time. This attribute is a place holder

- schedulingStatusAsString

```
virtual const char* schedulingStatusAsString() const;
```

Return my scheduling status as a character string.

- schedulingStatus

```
virtual int schedulingStatus(enum  
PlRpAcDpr::PlRpAcDprSchedulingStatus );
```

Set my scheduling status

- schedulingStatus

```
virtual int schedulingStatus(const char* aSchedulingStatus);
```

Set my status with a character string of SCHEDUABLE or UNSCHEDUABLE Return TRUE/
FALSE if the set worked or not.

- schedulingStatus

```
virtual enum PlRpAcDpr::PlRpAcDprSchedulingStatus  
schedulingStatus() const;
```

Get my scheduling status

- statusAsString

```
virtual const char* statusAsString() const;
```

Return my status as a character string.

- status

```
virtual int status(enum PlRpAcDpr::PlRpAcDprStatus );
```

Set my overall status

- status

```
virtual enum PlRpAcDpr::PlRpAcDprStatus status() const;
```

Get my overall status

- status

```
virtual int status(const char* aStatus);
```

Set my status with a character string of VALID or INVALID Return TRUE/FALSE if the set
worked or not.

- stringId

```
virtual int stringId(const char* aStringId);
```

Set my String ID

- stringId

```
virtual const HString& stringId() const;
```

Get my String ID

- taskLevel

```
virtual int taskLevel(const HEpochTime& aTaskLevel);
```

Set my task level

- taskLevel

```
virtual const HEpochTime& taskLevel() const;
```

Get my task level

- xdr

```
virtual int xdr(XDR* xdrs);
```

Get/put myself onto an XDR stream

6.3.15.3.3 PIRpAcForceLink Class

Overview:

class PIRpAcForceLink (container class for dummy class instantiation)

Instances of PIRpAcForceLink are dummy classes, used to trick the linker into linking classes which are usually not known at compile time. These are objects which might be dynamically loaded off of a stream.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PIRpAcForceLink();
```

Default constructor

```
PIRpAcForceLink(const PIRpAcForceLink& orig);
```

Copy constructor

```
virtual ~PIRpAcForceLink();
```

Destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpAcForceLink );
```

class PIRpAcForceLink (container class for dummy class instantiation)

Instances of PIRpAcForceLink are dummy classes, used to trick the linker into linking classes which are usually not known at compile time. These are objects which might be dynamically loaded off of a stream.

- linkage

```
virtual void linkage();
```

Protocol for forcing linkage of class objects not known at run time. Create dummy objects of all the types which you desire to force into the link. This behavior is never actually called at run time, just here to force the compiler to link classes in.

- operator =

```
PIRpAcForceLink& operator =(const PIRpAcForceLink& orig);
```

Assignment operator

6.3.15.3.4 PIRpAcGroundEvent Class

Overview:

Instances of PIRpAcGroundEvent are activities which model ground events on resources.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpAcActivity

Attributes:

Constructors and Destructor:

```
PIRpAcGroundEvent ( );
```

Default constructor

```
PIRpAcGroundEvent(const PIRpAcGroundEvent& orig);
```

Copy constructor

```
virtual ~PIRpAcGroundEvent ( );
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpAcGroundEvent );
```

Instances of PIRpAcGroundEvent are activities which model ground events on resources.

- get

```
virtual int get(istream& istr);
```

Get myself onto a stream

- operator =

```
PIRpAcGroundEvent& operator =(const PIRpAcGroundEvent& orig);
```

Assignment operator

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Get/put myself onto an XDR stream

6.3.15.3.5 PIRpAcInterval Class

Overview:

Class PIRpAcInterval is derived from Class HEpochInterval

This class defines attributes and operations [get/set] for PIRpAcInterval. One instance of this class presents one time slot. If myOnOff is FALSE, time slot is unselected. Otherwise, time slot is selected.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myOnOff: int
```

This interval is used for scheduling resources or temporary place for unschedulable resource Id

Constructors and Destructor:

```
PlRpAcInterval(const HEpochTime& sttTime, const HEpochTime&
stpTime);
```

Constructor with given starttime and stop time, by default myOnOff is TRUE

```
PlRpAcInterval(const PlRpAcInterval& orig);
```

Copy constructor

```
PlRpAcInterval();
```

Default constructor

```
virtual ~PlRpAcInterval();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpAcInterval );
```

Class PlRpAcInterval is derived from Class HEpochInterval

This class defines attributes and operations [get/set] for PlRpAcInterval. One instance of this class presents one time slot. If myOnOff is FALSE, time slot is unselected. Otherwise, time slot is selected.

- get

```
virtual int get(istream& istr);
```

Get myself from an ostream

- onOff

```
virtual int onOff(int );
```

Set on off value, input must be TRUE or FALSE

- onOff

```
virtual int onOff() const;
```

Get on off value

- operator =

```
PlRpAcInterval& operator =(const PlRpAcInterval& orig);
```

Assignment operator

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto an ostream

- unschedulableRsId

```
virtual void unschedulableRsId(int );
```

Set unschedulable resource id

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get myself onto XDR stream

6.3.15.3.6 PIRpAcResourceReservation Class

Overview:

This class defines attributes and operations for PIRpAcResourceReservation. Time interval list is generated when new reservation is registered to the reservation table

Export Control: Public

Inheritance Relationships:

Inherits from PIRpAcActivity

Attributes:

```
myActivity: int
```

Activity type

```
myConflictIntervalList: HEpochIntervalList
```

Conflict Interval List

```
myDescription: HString
```

Description

```
myEditDate: HDateTime
```

Edit Date

```
myEveryn: int
```

Everyn

```
myExclusive: int
```

Exclusive

```
myFrequency: enum PIRpAcResourceReservation::PIRpAcFrequency
```

Frequency type

myIntervalList: HEpochIntervalList

Interval list

myOriginator: HString

Originator

myRsList: HVList

Resource list

mySponsor: HString

Sponsor

myStartDateTime: HDateTime

Start Date/Time

myStartDate: HDate

Start Date

myStartTime: HTime

Start Time

myState: enum PlRpAcResourceReservation::PlRpAcState

State

myStopDateTime: HDateTime

Stop Date/Time

myStopDate: HDate

Stop Date

myStopTime: HTime

Stop Time

Constructors and Destructor:

```
PlRpAcResourceReservation();
```

Default constructor

```
PlRpAcResourceReservation(const PlRpAcResourceReservation&  
orig);
```

Copy constructor

```
virtual ~PlRpAcResourceReservation();
```

Destructor

Operations:

- activityId

```
virtual int activityId() const;
```

Get activity id.

- activityId

```
virtual void activityId(int );
```

Set activity id.

- addConflictIntervalList

```
virtual void addConflictIntervalList(PlRpAcInterval* );
```

Set conflicted interval into conflicted intervals list

- addIntervalList

```
virtual void addIntervalList(PlRpAcInterval* );
```

Set interval into intervals list

- addRs

```
virtual void addRs(int );
```

Set selected resource id into resources list

- cleanup

```
virtual void cleanup();
```

Clean up all lists

- conflictIntervalList

```
virtual HObjIter* conflictIntervalList();
```

Get pointer of conflicted interval list

- conflictIntervalSize

```
virtual int conflictIntervalSize();
```

Get conflicted interval list's size

- copyOverWithoutRActAll

```
virtual int copyOverWithoutRActAll(const  
PlRpAcResourceReservation& orig);
```

Copy over without RActAll from RSimpleAct

- copyOver

```
virtual int copyOver(const PIRpAcResourceReservation& );
```

Copy internals of source reservation as PIRpAcResourceReservation

- copyOver

```
virtual int copyOver(const RSimpleAct& );
```

Copy internals of source reservation as RSimpleAct

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpAcResourceReservation );
```

This class defines attributes and operations for PIRpAcResourceReservation. Time interval list is generated when new reservation is registered to the reservation table

- deleteLists

```
virtual void deleteLists();
```

Delete resource and interval lists

- description

```
virtual const HString& description() const;
```

Get description

- description

```
virtual void description(const char* aDescription);
```

Set description

- editDate

```
virtual const HDateTime& editDate() const;
```

Get edit date

- editDate

```
virtual void editDate(const HDateTime& aEditDate);
```

Set edit date

- everyn

```
virtual void everyn(int );
```

Set everyn

- everyn

```
virtual int everyn() const;
```

Get everyn

- exclusive

```
virtual void exclusive(int );
```

Set exclusive

- exclusive

```
virtual int exclusive() const;
```

Get exclusive

- explode

```
virtual int explode(const HDate& startDate, const HTime&
startTime, const HTime& stopTime);
```

Populate an interval

- explode

```
virtual int explode();
```

Populate interval list base on startDate, stopDate, and frequency

- explode

```
virtual int explode(const HDate& startDate, const HTime&
startTime, const HTime& stopTime, const HDate& stopDate, int*
days);
```

Populate intervals base on onOff in the weekly period

- explode

```
virtual int explode(const HDate& startDate, const HTime&
startTime, const HTime& stopTime, const HDate& stopDate, const
int repeat);
```

Populate intervals base on everyn

- explode

```
virtual int explode(const HDate& startDate, const HTime&
startTime, const HTime& stopTime, const HDate& stopDate, enum
PlRpAcResourceReservation::PlRpAcFrequency amonth);
```

Populate intervals base on monthy

- frequencyAsString

```
virtual const char* frequencyAsString() const;
```

Get frequency as string

- frequency

```
virtual enum PlRpAcResourceReservation::PlRpAcFrequency  
frequency() const;
```

Get frequency as frequency type

- frequency

```
virtual void frequency(enum  
PlRpAcResourceReservation::PlRpAcFrequency aFrequency);
```

Set frequency by given frequency type

- frequency

```
virtual int frequency(const char* aFrequency);
```

Set frequency by given string

- get

```
virtual int get(istream& istr);
```

Get myself onto a stream

- insertNewIntervalList

```
virtual int insertNewIntervalList(HObjCollection& col);
```

Insert new interval list

- insertNewRsList

```
virtual int insertNewRsList(HVList& newRsList);
```

Insert new resources list

- intervalList

```
virtual HObjIter* intervalList();
```

Get pointer of interval list

- intervalSize

```
virtual int intervalSize();
```

Get interval list's size

- operator =

```
PlRpAcResourceReservation& operator =(const  
PlRpAcResourceReservation& orig);
```

Assignment operator

- originator

```
virtual void originator(const char* aOriginator);
```

Set originator

- originator

```
virtual const HString& originator() const;
```

Get originator

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- removeAnInterval

```
virtual int removeAnInterval(PlRpAcInterval* );
```

return TRUE if remove an interval successfully from interval list

- retrieveAnConflictInterval

```
virtual PlRpAcInterval*
```

```
retrieveAnConflictInterval(PlRpAcInterval* aInterval);
```

Get interval from conflicted interval list

- retrieveAnInterval

```
virtual PlRpAcInterval* retrieveAnInterval(PlRpAcInterval*  
aInterval);
```

Get interval from interval list

- rsList

```
virtual HVList rsList();
```

Get resource list

- rsSize

```
virtual int rsSize();
```

Get resource list's size

- sponsor

```
virtual void sponsor(const char* aSponsor);
```

Set sponsor

- sponsor

```
virtual const HString& sponsor() const;
```

Get sponsor

- startDateTime

```
virtual const HDateTime& startDateTime() const;
```

Get start Date/Time

- startDateTime

```
virtual void startDateTime(const HDateTime& aStartDateTime);
```

Set start Date/Time

- startDate

```
virtual void startDate(const HDate& aStartDate);
```

Set start date

- startTime

```
virtual void startTime(const HTime& aStartTime);
```

Set start time

- stateAsString

```
virtual const char* stateAsString() const;
```

Get state as string

- stateTransitionTo

```
virtual int stateTransitionTo(enum  
PlRpAcResourceReservation::PlRpAcState aState);
```

Change status of reservation

- state

```
virtual enum PlRpAcResourceReservation::PlRpAcState state()  
const;
```

Get state as status type

- state

```
virtual void state(enum PlRpAcResourceReservation::PlRpAcState  
);
```

Set state by given status type

- state

```
virtual int state(const char* aState);
```

Set state by given string

- stopDateTime

```
virtual const HDateTime& stopDateTime() const;
```

Get stop Date/Time

- stopDateTime

```
virtual void stopDateTime(const HDateTime& aStopDateTime);
```

Set stop Date/Time

- stopDate

```
virtual void stopDate(const HDate& aStopDate);
```

Set stop date

- stopTime

```
virtual void stopTime(const HTime& aStopTime);
```

Set stop time

- xdr

```
virtual int xdr(XDR* xdrs);
```

Get/put myself onto an XDR stream

6.3.15.3.7 PIRpAcSchedInfo Class

Overview:

class PIRpAcSchedInfo (scheduling information base class)

Instances of PIRpAcSchedInfo are objects that contain information related to how an activity will/was scheduled.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myStartTimeInfo: PIRpAcTimeInfo*
```

My start time scheduling information

```
myStopTimeInfo: PIRpAcTimeInfo*
```

My stop time scheduling information

Constructors and Destructor:

```
PlRpAcSchedInfo();
```

Default constructor

```
PlRpAcSchedInfo(const PlRpAcSchedInfo& orig);
```

Copy constructor, I will do a deep copy making duplicates of the objects of the passed in class

```
virtual ~PlRpAcSchedInfo();
```

Destructor

Operations:

- cleanup

```
void cleanup();
```

Cleanup my internals

- copyOver

```
int copyOver(const PlRpAcSchedInfo& alloc);
```

Copy over my internals, non-virtual so derived classes can have their own

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpAcSchedInfo );
```

class PlRpAcSchedInfo (scheduling information base class)

Instances of PlRpAcSchedInfo are objects that contain information related to how an activity will/was scheduled.

- get

```
virtual int get(istream& istr);
```

Get myself onto a stream

- operator =

```
PlRpAcSchedInfo& operator =(const PlRpAcSchedInfo& orig);
```

Assignment operator, I will do a deep copy making duplicates of the objects of the passed in class

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- startMethodInfo

```
virtual PlRpAcTimeInfo* startTimeInfo();
```

Get my start time info

- startTimeInfo

```
virtual int startTimeInfo(PlRpAcTimeInfo* timeInfo);
```

Set my start time info, I assume ownership on the set and will delete my current start time info if I already have one

- stopTimeInfo

```
virtual int stopTimeInfo(PlRpAcTimeInfo* timeInfo);
```

Set my stop time info, I assume ownership on the set and will delete my current stop time info if I already have one

- stopTimeInfo

```
virtual PlRpAcTimeInfo* stopTimeInfo();
```

Get my stop time info

- xdr

```
virtual int xdr(XDR* xdrs);
```

Get/put myself onto an XDR stream

6.3.15.3.8 PlRpAcTimeInfo Class

Overview:

class PlRpAcTimeInfo (time related activity scheduling information)

Instances of PlRpAcTimeInfo are base class objects for scheduling time information

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PlRpAcTimeInfo();
```

Default constructor

```
PlRpAcTimeInfo(const PlRpAcTimeInfo& orig);
```

Copy constructor

```
virtual ~PlRpAcTimeInfo();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpAcTimeInfo );
```

class PlRpAcTimeInfo (time related activity scheduling information)

Instances of PlRpAcTimeInfo are base class objects for scheduling time information

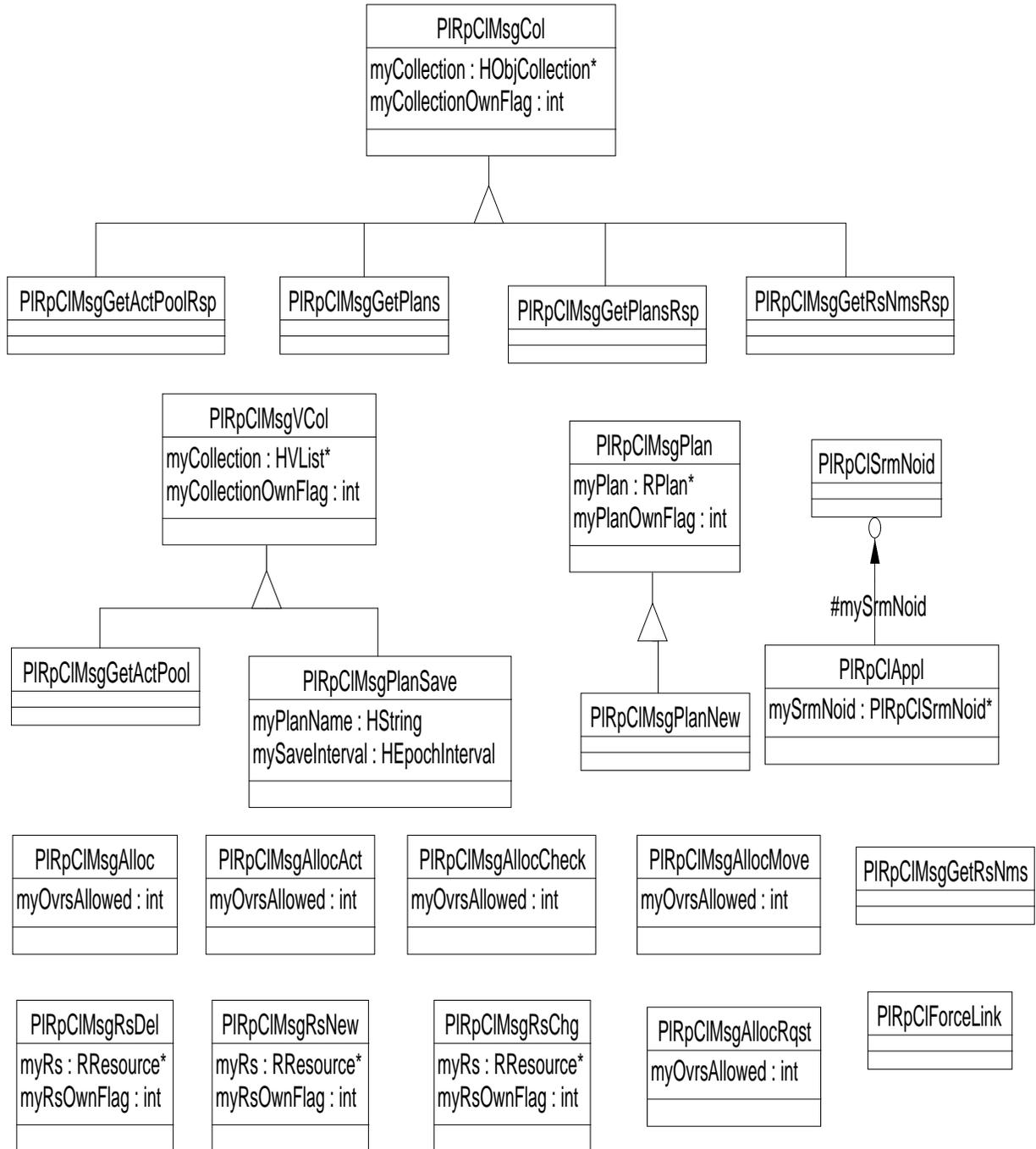
- operator =

```
PlRpAcTimeInfo& operator =(const PlRpAcTimeInfo& orig);
```

Assignment operator

6.3.16 Resource_Planning_CI Class Category

6.3.16.1 Overview



6.3.16.2 Resource_Planning_CI Classes

6.3.16.3 PIRpCIAppl Class

Overview:

A PIRpCIAppl is an ECS resource model client application class It is responsible for setting up communication to a resource model and initializing the entire process. PIRpCIAppl contains an srm noid for sending messages to the srm.

Export Control: Public

Inheritance Relationships:

Inherits from PIMiAppl

Attributes:

`mySrmNoid: PIRpClSrmNoid*`

My srm noid. It is owned by the agent and is therefore cleaned up there.

Constructors and Destructor:

```
PIRpCIAppl();
```

Default Constructor

```
PIRpCIAppl(const PIRpCIAppl& orig);
```

Copy Constructor

```
virtual ~PIRpCIAppl();
```

Destructor

Operations:

- connectionMade

```
virtual int connectionMade();
```

This member is called when the srm noid has made a connection to the resource model. Derived classes should override for any behavior that is to occur after a connection is made to the resource model.

- createSrmNoid

```
virtual int createSrmNoid();
```

Create the srm noid. This should be overridden in the base class.

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpClAppl );
```

A PIRpClAppl is an ECS resource model client application class It is responsible for setting up communication to a resource model and initializing the entire process. PIRpClAppl contains an srm noid for sending messages to the srm.

- initCommunications

```
virtual int initCommunications();
```

Initializes the srm noid, and connects to the srm.

- init

```
virtual int init();
```

This function provides protocol for initialization done here.

- newSrmNoid

```
virtual PIRpClSrmNoid* newSrmNoid();
```

Instantiate an srm noid. This should be overridden in derived classes to instantiate a derived type of PIRpClSrmNoid.

- operator =

```
PIRpClAppl& operator =(const PIRpClAppl& orig);
```

Assignment Operator

- printFlags

```
virtual void printFlags();
```

Dump a description of what flags are available and required to standard output.

6.3.16.3.1 PIRpClForceLink Class

Overview:

class PIRpClForceLink (force link for ECS resource model classes)

Instances of PIRpClForceLink are used to force linkage of ECS resource model message classes which may not be present at link time.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PIRpClForceLink();
```

Default constructor

```
PIRpClForceLink(const PIRpClForceLink& orig);
```

Copy constructor

```
virtual ~PIRpClForceLink();
```

Destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpClForceLink );
```

class PIRpClForceLink (force link for ECS resource model classes)

Instances of PIRpClForceLink are used to force linkage of ECS resource model message classes which may not be present at link time.

- linkage

```
void linkage();
```

Protocol for forcing linkage of class objects not known at run time. Create dummy objects of all the types which you desire to force into the link. This behavior is never actually called at run time, just here to force the compiler to link classes in.

- operator =

```
PIRpClForceLink& operator =(const PIRpClForceLink& orig);
```

Assignment operator

6.3.16.3.2 PIRpCIMsgAllocAct Class

Overview:

class PIRpCIMsgAllocAct (allocation message with activity, oversubscription).

PIRpCIMsgAllocAct is a request to allocate a specific resource to a specific activity on a specific plan over a specific interval. Constraints will be checked to determine if the allocation is feasible. The activity will be added to the RActPool. Oversubscription may be specified.

Export Control: Public

Inheritance Relationships:

Attributes:

`myOvrsAllowed: int`

Specifies whether or not oversubscription is allowed

Constructors and Destructor:

```
PlRpClMsgAllocAct(const PlRpClMsgAllocAct& orig);
```

Copy constructor.

```
PlRpClMsgAllocAct(const char* planName, const HEpochInterval&  
intvl, int rsId, RActivity* act, int ovrsAllowed);
```

Constructor taking member data. I do not assume ownership of the provided RActivity.

```
PlRpClMsgAllocAct();
```

Default constructor

```
virtual ~PlRpClMsgAllocAct();
```

Destructor. I will delete any data that I own.

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpClMsgAllocAct );
```

class PlRpClMsgAllocAct (allocation message with activity, oversubscription).

PlRpClMsgAllocAct is a request to allocate a specific resource to a specific activity on a specific plan over a specific interval. Constraints will be checked to determine if the allocation is feasible. The activity will be added to the RActPool. Oversubscription may be specified.

- get

```
virtual int get(istream& istr);
```

Get myself from a stream

- operator =

```
PlRpClMsgAllocAct& operator =(const PlRpClMsgAllocAct& orig);
```

Assignment operator. The data in the source message is cloned and owned.

- overSubAllowed

```
virtual int overSubAllowed(int yesNo);
```

Set whether oversubscription is allowed.

- overSubAllowed

```
virtual int overSubAllowed() const;
```

Get whether oversubscription is allowed.

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto an ostream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get myself from an XDR stream

6.3.16.3.3 PIRpCIMsgAllocCheck Class

Overview:

Class PIRpCIMsgAllocCheck (ecs check allocation message)

PIRpCIMsgAllocCheck is a SMsgAllocCheck request to check a specific resource to see if a specific activity on a specific plan over a specific interval can be allocated on that resource. This is determined by constraint-checking. The message also holds state to determine if oversubscription is allowed.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myOvrsAllowed: int
```

Specifies whether or not oversubscription is allowed

Constructors and Destructor:

```
PIRpCIMsgAllocCheck(const PIRpCIMsgAllocCheck& orig);
```

Copy constructor.

```
PIRpCIMsgAllocCheck(const char* planName, const HEpochInterval&  
intvl, int rsId, int actId, int ovrsAllowed);
```

Constructor taking member data.

```
PIRpCIMsgAllocCheck();
```

Default constructor.

```
virtual ~PIRpCIMsgAllocCheck();
```

Destructor.

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgAllocCheck );
```

Class PIRpCIMsgAllocCheck (ecs check allocation message)

PIRpCIMsgAllocCheck is a SMsgAllocCheck request to check a specific resource to see if a specific activity on a specific plan over a specific interval can be allocated on that resource. This is determined by constraint-checking. The message also holds state to determine if oversubscription is allowed.

- get

```
virtual int get(istream& istr);
```

Get myself from a stream

- operator =

```
PIRpCIMsgAllocCheck& operator =(const PIRpCIMsgAllocCheck&  
orig);
```

Assignment operator

- overSubAllowed

```
virtual int overSubAllowed(int yesNo);
```

Set whether oversubscription is allowed or not

- overSubAllowed

```
virtual int overSubAllowed() const;
```

Get whether oversubscription is allowed or not

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get myself from a XDR stream

6.3.16.3.4 PIRpCIMsgAllocMove Class

Overview:

Class PIRpCIMsgAllocMove (force allocation message)

PIRpCIMsgAllocMove is a request to allocate a specific resource to a specific activity on a specific plan over a specific interval. The allocation is forced - no constraints are checked. Information is added to specify if oversubscription is permitted or not.

Export Control: Public

Inheritance Relationships:

Attributes:

`myOvrsAllowed: int`

Specifies whether or not oversubscription is allowed

Constructors and Destructor:

```
PIRpCIMsgAllocMove(const PIRpCIMsgAllocMove& orig);
```

Copy constructor.

```
PIRpCIMsgAllocMove(int actId, int sourceRsId, int destRsId,  
const char* sourcePlanName, const char* destPlanName, const  
HEpochInterval& oldIntvl, const HEpochInterval& newIntvl, int  
ovrsAllowed, int impactFlag);
```

Constructor taking member data.

```
PIRpCIMsgAllocMove();
```

Default constructor.

```
virtual ~PIRpCIMsgAllocMove();
```

Destructor.

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgAllocMove );
```

Class PIRpCIMsgAllocMove (force allocation message)

PIRpCIMsgAllocMove is a request to allocate a specific resource to a specific activity on a specific plan over a specific interval. The allocation is forced - no constraints are checked. Information is added to specify if oversubscription is permitted or not.

- get

```
virtual int get(istream& istr);
```

Get myself from a stream

- operator =

```
PIRpCIMsgAllocMove& operator =(const PIRpCIMsgAllocMove& orig);
```

Assignment operator.

- overSubAllowed

```
virtual int overSubAllowed(int yesNo);
```

Set whether oversubscription is allowed or not

- overSubAllowed

```
virtual int overSubAllowed() const;
```

Get whether oversubscription is allowed or not

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get myself from a XDR stream

6.3.16.3.5 PIRpCIMsgAllocRqst Class

Overview:

class PIRpCIMsgAllocRqst (allocation message with activity,oversubscription)

PIRpCIMsgAllocRqst is a request to allocate a specific resource to a specific activity on a specific plan over a specific interval. Constraints will be checked to determine if the allocation is feasible. The activity will be added to the RActPool. Oversubscription may be specified.

Export Control: Public

Inheritance Relationships:

Attributes:

`myOvrsAllowed: int`

Specifies whether or not oversubscription is allowed

Constructors and Destructor:

```
PlRpClMsgAllocRqst(const PlRpClMsgAllocRqst& orig);
```

Copy constructor.

```
PlRpClMsgAllocRqst(const char* planName, HObjCollection*  
reqsToSch, HObjCollection* schRqsts, HObjCollection*  
notSchRqsts, int ovrsAllowed);
```

Constructor taking member data.

```
PlRpClMsgAllocRqst();
```

Default constructor

```
virtual ~PlRpClMsgAllocRqst();
```

Destructor. I will delete any data that I own.

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpClMsgAllocRqst );
```

class PIRpCIMsgAllocRqst (allocation message with activity,oversubscription)

PIRpCIMsgAllocRqst is a request to allocate a specific resource to a specific activity on a specific plan over a specific interval. Constraints will be checked to determine if the allocation is feasible. The activity will be added to the RActPool. Oversubscription may be specified.

- get

```
virtual int get(istream& istr);
```

Get myself from a stream

- operator =

```
PlRpClMsgAllocRqst& operator =(const PlRpClMsgAllocRqst& orig);
```

Assignment operator. The data in the source message is cloned and owned.

- overSubAllowed

```
virtual int overSubAllowed(int yesNo);
```

Set whether oversubscription is allowed.

- overSubAllowed

```
virtual int overSubAllowed() const;
```

Get whether oversubscription is allowed.

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get myself from a XDR stream

6.3.16.3.6 PIRpCIMsgAlloc Class

Overview:

Class PIRpCIMsgAlloc (allocation message).

PIRpCIMsgAlloc is a request to allocate a specific resource to a specific activity on a specific plan over a specific interval. Constraints will be checked to determine if the allocation is feasible. The user may specify whether oversubscription is permitted or not.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myOvrsAllowed: int
```

Specifies whether or not oversubscription is allowed

Constructors and Destructor:

```
PIRpCIMsgAlloc(const PIRpCIMsgAlloc& orig);
```

Copy constructor.

```
PIRpCIMsgAlloc(const char* planName, const HEpochInterval&  
intvl, int rsId, int actId, int ovrsAllowed);
```

Constructor taking member data.

```
PIRpCIMsgAlloc();
```

Default constructor

```
virtual ~PIRpCIMsgAlloc();
```

Destructor.

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgAlloc );
```

Class PIRpCIMsgAlloc (allocation message).

PIRpCIMsgAlloc is a request to allocate a specific resource to a specific activity on a specific plan over a specific interval. Constraints will be checked to determine if the allocation is feasible. The user may specify whether oversubscription is permitted or not.

- get

```
virtual int get(istream& istr);
```

Get myself from a stream

- operator =

```
PIRpCIMsgAlloc& operator =(const PIRpCIMsgAlloc& orig);
```

Assignment operator.

- overSubAllowed

```
virtual int overSubAllowed(int yesNo);
```

Set whether oversubscription is allowed or not

- overSubAllowed

```
virtual int overSubAllowed() const;
```

Get whether oversubscription is allowed or not

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto an ostream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get myself from an XDR stream

6.3.16.3.7 PIRpCIMsgCol Class

Overview:

class PIRpCIMsgCol (collection message with status)

Instances of PIRpClMsgCol are a message that carries collections from one agent to another.
NOTE: copy and assignment will result in a DEEP COPY of the messages carried collection I.E.
- the message copied to or assigned to will own its own copy of the data.

Export Control: Public

Inheritance Relationships:

Attributes:

myCollectionOwnFlag: int

Specifies whether or not I own myCollection

myCollection: HObjCollection*

The collection that I carry

Constructors and Destructor:

PIRpClMsgCol(const PIRpClMsgCol& orig);

Copy constructor

PIRpClMsgCol(HObjCollection* col);

Constructor taking a pointer to a collection. I do not assume ownership of the passed in collection.

PIRpClMsgCol();

Default constructor

virtual ~PIRpClMsgCol();

Destructor

Operations:

- cleanup

void cleanup();

Clean up any data members

- col

virtual HObjCollection* col();

Get my collection. I will relinquish ownership on the get.

- col

```
virtual int col(HObjCollection* col);
```

Set my collection. I do not assume ownership on the set.

- copyOver

```
int copyOver(const PlRpClMsgCol& orig);
```

Copy over data from another PlRpClMsgCol

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpClMsgCol );
```

class PlRpClMsgCol (collection message with status)

Instances of PlRpClMsgCol are a message that carries collections from one agent to another.

NOTE: copy and assignment will result in a DEEP COPY of the messages carried collection I.E.

- the message copied to or assigned to will own its own copy of the data.

- get

```
virtual int get(istream& istr);
```

Get myself from a stream

- operator =

```
PlRpClMsgCol& operator =(const PlRpClMsgCol& orig);
```

Assignment operator

- ownData

```
virtual int ownData();
```

Ask me to own my data

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- relData

```
virtual int relData();
```

Ask me to relinquish ownership of my data

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get myself from a XDR stream

6.3.16.3.8 PIRpCIMsgGetActPoolRsp Class

Overview:

Comprises a response to a request for activities, and carries a copy of all or part of the resource models activity pool.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpCIMsgCol

Attributes:

Constructors and Destructor:

```
PIRpCIMsgGetActPoolRsp(const PIRpCIMsgGetActPoolRsp& orig);
```

Copy constructor

```
PIRpCIMsgGetActPoolRsp(HObjCollection* actPool);
```

I will not assume ownership of the given collection

```
PIRpCIMsgGetActPoolRsp();
```

Default constructor

```
virtual ~PIRpCIMsgGetActPoolRsp();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgGetActPoolRsp );
```

Comprises a response to a request for activities, and carries a copy of all or part of the resource models activity pool.

- operator =

```
PIRpCIMsgGetActPoolRsp& operator =(const PIRpCIMsgGetActPoolRsp& orig);
```

Assignment operator

6.3.16.3.9 PIRpCIMsgGetActPool Class

Overview:

Instances of PIRpCIMsgGetActPool are messages requesting activities from the resource model. The carried HVList should be a list of integer activity ID's. The corresponding activity to each ID will be returned in the response. If no list is supplied, all activities known to the resource model will be returned.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpCIMsgVCol

Attributes:

Constructors and Destructor:

```
PlRpClMsgGetActPool(const PlRpClMsgGetActPool& orig);
```

Copy constructor

```
PlRpClMsgGetActPool(HVList* actIdList);
```

Constructor taking member data

```
PlRpClMsgGetActPool();
```

Default constructor

```
virtual ~PlRpClMsgGetActPool();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpClMsgGetActPool );
```

Instances of PIRpCIMsgGetActPool are messages requesting activities from the resource model. The carried HVList should be a list of integer activity ID's. The corresponding activity to each ID will be returned in the response. If no list is supplied, all activities known to the resource model will be returned.

- operator =

```
PlRpClMsgGetActPool& operator =(const PlRpClMsgGetActPool& orig);
```

Assignment operator

6.3.16.3.10 PIRpCIMsgGetPlansRsp Class

Overview:

class PIRpCIMsgGetPlansRsp (Plans message)

Instances of PIRpCIMsgGetPlansRsp are a response to a request sent by an PIRpCIMsgGetPlans. PIRpCIMsgGetPlansRsp carries a collection of plans corresponding to the plan names carried in the PIRpCIMsgGetPlans. If no names were specified, then the response contains all plans in the resource model.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpCIMsgCol

Attributes:

Constructors and Destructor:

```
PIRpCIMsgGetPlansRsp(const PIRpCIMsgGetPlansRsp& orig);
```

Copy constructor

```
PIRpCIMsgGetPlansRsp(HObjCollection* planList);
```

Constructor taking a collection of plans I do not assume ownership of the passed in collection.

```
PIRpCIMsgGetPlansRsp();
```

Default constructor

```
virtual ~PIRpCIMsgGetPlansRsp();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgGetPlansRsp );
```

class PIRpCIMsgGetPlansRsp (Plans message)

Instances of PIRpCIMsgGetPlansRsp are a response to a request sent by an PIRpCIMsgGetPlans.

PIRpCIMsgGetPlansRsp carries a collection of plans corresponding to the plan names carried in the PIRpCIMsgGetPlans. If no names were specified, then the response contains all plans in the resource model.

- operator =

```
PIRpCIMsgGetPlansRsp& operator =(const PIRpCIMsgGetPlansRsp&
orig);
```

Assignment operator

6.3.16.3.11 PIRpCIMsgGetPlans Class

Overview:

class PIRpCIMsgGetPlans (Get plans message)

Instances of PIRpCIMsgGetPlans are a request to get a subset of plans from the resource model. All plans with names matching those given in the collection will be returned. If no names are specified, then all plans will be returned.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpCIMsgCol

Attributes:

Constructors and Destructor:

```
PIRpCIMsgGetPlans(const PIRpCIMsgGetPlans& orig);
```

Copy constructor

```
PIRpCIMsgGetPlans(HObjCollection* planNames);
```

Constructor taking a pointer to a collection of plan names. I do not assume ownership of the passed in collection.

```
PIRpCIMsgGetPlans();
```

Default constructor

```
virtual ~PIRpCIMsgGetPlans();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgGetPlans );
```

class PIRpCIMsgGetPlans (Get plans message)

Instances of PIRpCIMsgGetPlans are a request to get a subset of plans from the resource model. All plans with names matching those given in the collection will be returned. If no names are specified, then all plans will be returned.

- operator =

```
PIRpCIMsgGetPlans& operator =(const PIRpCIMsgGetPlans& orig);
```

Assignment operator

6.3.16.3.12 PIRpCIMsgGetRsNmsRsp Class

Overview:

Class PIRpCIMsgGetRsNmsRsp (resource names message)

PIRpCIMsgGetRsNmsRsp contains the names of all resources known by the resource model. It is sent in response to a request for resource names (PIRpCIMsgRsNms).

Export Control: Public

Inheritance Relationships:

Inherits from PIRpCIMsgCol

Attributes:

Constructors and Destructor:

```
PIRpCIMsgGetRsNmsRsp(const PIRpCIMsgGetRsNmsRsp& orig);
```

Copy constructor. The data in the source message is cloned and owned.

```
PIRpCIMsgGetRsNmsRsp(HObjCollection* nameList);
```

Constructor taking a list of resource names. I do not assume ownership of the list.

```
PIRpCIMsgGetRsNmsRsp();
```

Default constructor.

```
virtual ~PIRpCIMsgGetRsNmsRsp();
```

Destructor. I will delete any data that I own.

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgGetRsNmsRsp );
```

Class PIRpCIMsgGetRsNmsRsp (resource names message)

PIRpCIMsgGetRsNmsRsp contains the names of all resources known by the resource model. It is sent in response to a request for resource names (PIRpCIMsgRsNms).

- operator =

```
PIRpCIMsgGetRsNmsRsp& operator =(const PIRpCIMsgGetRsNmsRsp& orig);
```

Assignment operator. The data in the source message is cloned and owned.

6.3.16.3.13 PIRpCIMsgGetRsNms Class

Overview:

class PIRpCIMsgGetRsNms (get resource names message)

PIRpCIMsgGetRsNms is a request to get the names of all resources known by the resource model.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PIRpCIMsgGetRsNms ( );
```

Default constructor.

```
PIRpCIMsgGetRsNms(const PIRpCIMsgGetRsNms& orig);
```

Copy constructor.

```
virtual ~PIRpCIMsgGetRsNms ( );
```

Destructor.

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgGetRsNms );
```

class PIRpCIMsgGetRsNms (get resource names message)

PIRpCIMsgGetRsNms is a request to get the names of all resources known by the resource model.

- operator =

```
PIRpCIMsgGetRsNms& operator =(const PIRpCIMsgGetRsNms& orig);
```

Assignment operator.

6.3.16.3.14 PIRpCIMsgPlanNew Class

Overview:

class PIRpCIMsgPlanNew (New Plan message)

Instances of PIRpCIMsgPlanNew are a message that tells the resource model about a new plan that has been created.

PIRpCIMsgPlanNew is a request to add the new plan it carries to some or all resources, as well as to place the plan in the resource model.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpCIMsgPlan

Attributes:

Constructors and Destructor:

```
PIRpCIMsgPlanNew(const PIRpCIMsgPlanNew& orig);
```

Copy constructor

```
PIRpCIMsgPlanNew(RPlan* );
```

Constructor taking a pointer to a plan.

```
PIRpCIMsgPlanNew();
```

Default constructor

```
virtual ~PIRpCIMsgPlanNew();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgPlanNew );
```

class PIRpCIMsgPlanNew (New Plan message)

Instances of PIRpCIMsgPlanNew are a message that tells the resource model about a new plan that has been created.

PIRpCIMsgPlanNew is a request to add the new plan it carries to some or all resources, as well as to place the plan in the resource model.

- operator =

```
PIRpCIMsgPlanNew& operator =(const PIRpCIMsgPlanNew& orig);
```

Assignment operator

6.3.16.3.15 PIRpCIMsgPlanSave Class

Overview:

class PIRpCIMsgPlanSave (save a plan to a file)

PIRpCIMsgPlanSave is a request to save a plan to a file whose name is specified at the command line. If a list of resource Ids is given, then only those resources will be saved, otherwise all of the resources on the specified plan will be saved. In addition, an interval may be specified over which to save. If no interval is specified, then the plan will be saved over all time.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpCIMsgVCol

Attributes:

myPlanName: HString

The name of the plan that is to be saved

mySaveInterval: HEpochInterval

The interval over which the plan should be saved (all time will be saved if the interval is not specified)

Constructors and Destructor:

```
PIRpCIMsgPlanSave(const PIRpCIMsgPlanSave& orig);
```

Copy constructor

```
PIRpCIMsgPlanSave(HVList* rsIdList);
```

Constructor taking member data

```
PIRpCIMsgPlanSave();
```

Default constructor

```
virtual ~PlRpClMsgPlanSave();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpClMsgPlanSave );
```

class PlRpClMsgPlanSave (save a plan to a file)

PlRpClMsgPlanSave is a request to save a plan to a file whose name is specified at the command line. If a list of resource Ids is given, then only those resources will be saved, otherwise all of the resources on the specified plan will be saved. In addition, an interval may be specified over which to save. If no interval is specified, then the plan will be saved over all time.

- get

```
virtual int get(istream& istr);
```

Get myself from a stream

- operator =

```
PlRpClMsgPlanSave& operator =(const PlRpClMsgPlanSave& orig);
```

Assignment operator

- planName

```
const HString& planName() const;
```

Get my plan name

- planName

```
int planName(const char* name);
```

Set my plan name

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- saveInterval

```
int saveInterval(const HEpochInterval& );
```

Set my save interval

- saveInterval

```
const HEpochInterval& saveInterval() const;
```

Get my save interval

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get myself from a XDR stream

6.3.16.3.16 PIRpCIMsgPlan Class

Overview:

class PIRpCIMsgPlan (Message carrying a plan)

Instances of PIRpCIMsgPlan are messages that have an associated plan. derived classes can add any additional information necessary to do something with the plan, such as add a new plan, or update an existing plan.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myPlanOwnFlag: int
```

Whether I own my data or not

```
myPlan: RPlan*
```

My plan

Constructors and Destructor:

```
PIRpCIMsgPlan(const PIRpCIMsgPlan& orig);
```

Restricted access to copy and assignment. Copy constructor The data in the source message is cloned and owned.

```
PIRpCIMsgPlan(RPlan* );
```

Constructor taking a pointer to a plan. I do not assume ownership of the passed in plan.

```
PIRpCIMsgPlan();
```

Default constructor

```
virtual ~PIRpCIMsgPlan();
```

Destructor I will delete any data that I own.

Operations:

- cleanup

```
virtual void cleanup();
```

Clean up any data that I own.

- copyOver

```
int copyOver(const PIRpCIMsgPlan& orig);
```

Clone and own the data in the source message.

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgPlan );
```

class PIRpCIMsgPlan (Message carrying a plan)

Instances of PIRpCIMsgPlan are messages that have an associated plan. derived classes can add any additional information necessary to do something with the plan, such as add a new plan, or update an existing plan.

- getPlan

```
virtual RPlan* getPlan();
```

Get my plan. I will relinquish ownership on the get.

- get

```
virtual int get(istream& istr);
```

Get myself from a stream

- operator =

```
PIRpCIMsgPlan& operator =(const PIRpCIMsgPlan& orig);
```

Assignment operator The data in the source message is cloned and owned.

- ownData

```
virtual int ownData();
```

Ask me to own my data

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- relData

```
virtual int relData();
```

Ask me to relinquish ownership of my data

- setPlan

```
virtual int setPlan(RPlan* );
```

Set my plan. I do not assume ownership on the set.

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get myself from a XDR stream

6.3.16.3.17 PIRpCIMsgRsChg Class

Overview:

class PIRpCIMsgRsChg (Modified Resource message)

Instances of PIRpCIMsgRsChg are a message that tells the resource model about a modified resource.

PIRpCIMsgRsChg is a request to modify the resource it carries in the resource model.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myRsOwnFlag: int
```

Specifies whether I own the resource object

```
myRs: RResource*
```

My resource object

Constructors and Destructor:

```
PIRpCIMsgRsChg(const PIRpCIMsgRsChg& orig);
```

Copy constructor

```
PIRpCIMsgRsChg(RResource* );
```

Constructor taking a pointer to a resource.

```
PIRpCIMsgRsChg( );
```

Default constructor

```
virtual ~PIRpClMsgRsChg();
```

Destructor

Operations:

- cleanup

```
virtual void cleanup();
```

Cleanup local attributes

- copyOver

```
int copyOver(const PIRpClMsgRsChg& orig);
```

Clone and own the data in the source message.

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpClMsgRsChg );
```

class PIRpClMsgRsChg (Modified Resource message)

Instances of PIRpClMsgRsChg are a message that tells the resource model about a modified resource.

PIRpClMsgRsChg is a request to modify the resource it carries in the resource model.

- get

```
virtual int get(istream& );
```

Get myself from a stream

- operator =

```
PIRpClMsgRsChg& operator =(const PIRpClMsgRsChg& orig);
```

Assignment operator

- ownData

```
virtual int ownData();
```

Ask me to own my data.

- put

```
virtual int put(ostream& ) const;
```

Put myself onto a stream

- relData

```
virtual int relData();
```

Ask me to release my data.

- rs

```
virtual RResource* rs();
```

Get the resource. If I previously assumed ownership, I will relinquish it.

- rs

```
virtual int rs(RResource* );
```

Set the resource. I do not assume ownership.

- xdr

```
virtual int xdr(XDR* );
```

Set/get myself from a XDR stream

6.3.16.3.18 PIRpCIMsgRsDel Class

Overview:

class PIRpCIMsgRsDel (Deleted Resource message)

Instances of PIRpCIMsgRsDel are a message that tells the resource model about a deleted resource.

PIRpCIMsgRsDel is a request to delete the resource it carries in the resource model.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myRsOwnFlag: int
```

Specifies whether I own the resource object

```
myRs: RResource*
```

My resource object

Constructors and Destructor:

```
PIRpCIMsgRsDel(const PIRpCIMsgRsDel& orig);
```

Copy constructor

```
PIRpCIMsgRsDel(RResource* );
```

Constructor taking a pointer to a resource.

```
PIRpCIMsgRsDel();
```

Default constructor

```
virtual ~PIRpCIMsgRsDel();
```

Destructor

Operations:

- cleanup

```
virtual void cleanup();
```

Cleanup local attributes

- copyOver

```
int copyOver(const PIRpCIMsgRsDel& orig);
```

Clone and own the data in the source message.

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgRsDel );
```

class PIRpCIMsgRsDel (Deleted Resource message)

Instances of PIRpCIMsgRsChg are a message that tells the resource model about a deleted resource.

PIRpCIMsgRsDel is a request to delete the resource it carries in the resource model.

- get

```
virtual int get(istream& );
```

Get myself from a stream

- operator =

```
PIRpCIMsgRsDel& operator =(const PIRpCIMsgRsDel& orig);
```

Assignment operator

- ownData

```
virtual int ownData();
```

Ask me to own my data.

- put

```
virtual int put(ostream& ) const;
```

Put myself onto a stream

- relData

```
virtual int relData();
```

Ask me to release my data.

- rs

```
virtual RResource* rs();
```

Get the resource. If I previously assumed ownership, I will relinquish it.

- rs

```
virtual int rs(RResource* );
```

Set the resource. I do not assume ownership.

- xdr

```
virtual int xdr(XDR* );
```

Set/get myself from a XDR stream

6.3.16.3.19 PIRpCIMsgRsNew Class

Overview:

class PIRpCIMsgRsNew (New Resource message)

Instances of PIRpCIMsgRsNew are a message that tells the resource model about a new resource.

PIRpCIMsgRsNew is a request to add a new resource to the resource model.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myRsOwnFlag: int
```

Specifies whether I own the resource object

```
myRs: RResource*
```

My resource object

Constructors and Destructor:

```
PIRpCIMsgRsNew(const PIRpCIMsgRsNew& orig);
```

Copy constructor

```
PIRpCIMsgRsNew(RResource* );
```

Constructor taking a pointer to a resource.

```
PIRpCIMsgRsNew( );
```

Default constructor

```
virtual ~PIRpCIMsgRsNew( );
```

Destructor

Operations:

- cleanup

```
virtual void cleanup();
```

Cleanup local attributes

- copyOver

```
int copyOver(const PIRpCIMsgRsNew& orig);
```

Clone and own the data in the source message.

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgRsNew );
```

class PIRpCIMsgRsNew (New Resource message)

Instances of PIRpCIMsgRsNew are a message that tells the resource model about a new resource.

PIRpCIMsgRsNew is a request to add a new resource to the resource model.

- get

```
virtual int get(istream& );
```

Get myself from a stream

- operator =

```
PIRpCIMsgRsNew& operator =(const PIRpCIMsgRsNew& orig);
```

Assignment operator

- ownData

```
virtual int ownData();
```

Ask me to own my data.

- put

```
virtual int put(ostream& ) const;
```

Put myself onto a stream

- relData

```
virtual int relData();
```

Ask me to release my data.

- rs

```
virtual RResource* rs();
```

Get the resource. If I previously assumed ownership, I will relinquish it.

- rs

```
virtual int rs(RResource* );
```

Set the resource. I do not assume ownership.

- xdr

```
virtual int xdr(XDR* );
```

Set/get myself from a XDR stream

6.3.16.3.20 PIRpCIMsgVCol Class

Overview:

class PIRpCIMsgVCol (collection message with status)

Instances of PIRpCIMsgVCol are a message that carries void collections from one agent to another. These void collections are ONLY to be used for sending lists of integer values. Any pointers in the list will be treated as an integer, and what they point at will not be sent. NOTE: copy and assignment will result in a DEEP COPY of the messages carried collection I.E. - the message copied to or assigned to will own its own copy of the data.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myCollectionOwnFlag: int
```

Specifies whether or not I own myCollection

```
myCollection: HVList*
```

The collection of integers I am to carry

Constructors and Destructor:

```
PIRpCIMsgVCol(const PIRpCIMsgVCol& orig);
```

Copy constructor

```
PIRpCIMsgVCol(HVList* list);
```

Constructor taking a pointer to a collection. I do not assume ownership of the passed in collection.

```
PIRpCIMsgVCol();
```

Default constructor

```
virtual ~PIRpCIMsgVCol();
```

Destructor

Operations:

- cleanup

```
void cleanup();
```

Clean up my data

- col

```
virtual HVList* col();
```

Get my collection. I will relinquish ownership on the get.

- col

```
virtual int col(HVList* list);
```

Set my collection. I do not assume ownership on the set.

- copyOver

```
int copyOver(const PIRpCIMsgVCol& orig);
```

Copy over data from the source PIRpCIMsgVCol

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpCIMsgVCol );
```

class PIRpCIMsgVCol (collection message with status)

Instances of PIRpCIMsgVCol are a message that carries void collections from one agent to another. These void collections are ONLY to be used for sending lists of integer values. Any pointers in the list will be treated as an integer, and what they point at will not be sent. NOTE: copy and assignment will result in a DEEP COPY of the messages carried collection I.E. - the message copied to or assigned to will own its own copy of the data.

- get

```
virtual int get(istream& istr);
```

Get myself from a stream

- operator =

```
PlRpClMsgVCol& operator =(const PlRpClMsgVCol& orig);
```

Assignment operator

- ownData

```
virtual int ownData();
```

Ask me to own my data

- put

```
virtual int put(ostream& ostr) const;
```

Put myself onto a stream

- relData

```
virtual int relData();
```

Ask me to relinquish ownership of my data

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get myself from a XDR stream

6.3.16.3.21 PIRpClSrmNoid Class

Overview:

class PIRpClSrmNoid (ecs resource model client)

Instances of PIRpClSrmNoid are HNoids that manage the finding, requests to, and receiving information from the system srm within the era process. They add additional behavior over the base SrmCliNoid, as well as adding additional information having to do with requesting allocations that may be oversubscribed.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PlRpClSrmNoid();
```

Default constructor

```
PlRpClSrmNoid(const PlRpClSrmNoid& orig);
```

Copy constructor

```
virtual ~PlRpClSrmNoid();
```

Destructor.

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpClSrmNoid );
```

class PIRpClSrmNoid (ecs resource model client)

Instances of PIRpClSrmNoid are HNoids that manage the finding, requests to, and receiving information from the system srm within the era process. They add additional behavior over the base SrmCliNoid, as well as adding additional information having to do with requesting allocations that may be oversubscribed.

- establishInterest

```
virtual int establishInterest();
```

Establish interest with my agent in the get Activity pool response

- handleActPoolRsp

```
virtual int handleActPoolRsp(HMsgConn* , HAddress* ,  
PlRpClMsgGetActPoolRsp* msg);
```

Method to handle incoming PIRpClMsgGetActPoolRsp messages. Sets this processes' Activity Pool to be the same as the incoming messages' pool

- handleMessage

```
virtual int handleMessage(HMsgConn* curCon, HAddress* curAdr,  
HMessage* curMsg);
```

Method to handle current inbound messages

- handlePlansGetRsp

```
virtual int handlePlansGetRsp(HMsgConn* , HAddress* ,  
PlRpClMsgGetPlansRsp* msg);
```

Add access plan to plan pool

- handlePIRpCIMsgPlanNew

```
virtual int handlePlRpClMsgPlanNew(HMsgConn* , HAddress* ,  
PlRpClMsgPlanNew* msg);
```

Add new plan to pool

- handleRsChg

```
virtual int handleRsChg(HMsgConn* currCon, HAddress* currAdr,  
PlRpClMsgRsChg* currMsg);
```

Handle changing a resource in the pool

- handleRsDel

```
virtual int handleRsDel(HMsgConn* currCon, HAddress* currAdr,  
PlRpClMsgRsDel* currMsg);
```

Handle deleting a resource from the pool

- handleRsGetNmsRsp

```
virtual int handleRsGetNmsRsp(HMsgConn* , HAddress* ,  
PlRpClMsgGetRsNmsRsp* msg);
```

Handle a message containing the names of all the resources known by the SRM. Default behavior is to just see if the request worked.

- handleRsNew

```
virtual int handleRsNew(HMsgConn* currCon, HAddress* currAdr,  
PlRpClMsgRsNew* currMsg);
```

Handle adding a resource to the pool

- installActNew

```
virtual int installActNew(RActivity* act);
```

Overridden from base class to install this activity into my local activity pool, if there is one and if the activity has a parent id, the activity will be added to the parent as well as the pool. This routine assumes ownership of the passed in RActivity. The default behavior is to add this RActivity to my RActPool, and return TRUE. If there is no RActPool or the RActivity cannot be added to the pool or there is already an RActivity with the given activity id, then delete the RActivity and return FALSE.

- installActPool

```
virtual int installActPool(HObjCollection* col);
```

Install the given collection into the local activity pool. Returns FALSE if the addition of any activity to the activity pool fails.

- installPlanNew

```
virtual int installPlanNew(const char* name, HVList* rsIds);
```

Install a new, empty plan given a plan name and an optional list of resource IDs. If no resource IDs are supplied, then the plan is installed against all RResources in the RRsPool. If a given RResource already has a plan of this name, then that RResource is skipped. This behavior does not presume ownership of the passed in HVList.

- installRsChg

```
virtual int installRsChg(RResource* rs);
```

Change a resource in the pool

- installRsDel

```
virtual int installRsDel(int rsId);
```

Remove a resource from the pool

- installRsNew

```
virtual int installRsNew(RResource* rs);
```

Add a resource to the pool

- operator =

```
PlRpClSrmNoid& operator =(const PlRpClSrmNoid& orig);
```

Assignment operator

- requestActPool

```
virtual int requestActPool(HVList* theCol);
```

Request to receive a copy of the SRM's activity pool

- requestAllocAct

```
virtual int requestAllocAct(const char* planName, const  
HEpochInterval& intvl, int rsId, RActivity* act, int ovrAllowed,  
const char* allocatorType);
```

Request that an allocation be performed, with constraint-checking. against an RActivity object. This is overloaded from my base class to permit the caller to specify if oversubscription is allowed or not.

- requestAllocAct

```
virtual int requestAllocAct(const char* planName, const
HEpochInterval& intvl, int rsId, RActivity* act, const char*
allocatorType);
```

Request that an allocation be performed, with constraint-checking. against an RActivity object. This is overridden from my base class to presume that oversubscription is not allowed.

- requestAllocCheck

```
virtual int requestAllocCheck(const char* planName, const
HEpochInterval& intvl, int rsId, int actId, int ovrAllowed);
```

Request that an allocation be checked. This will determine if the allocation COULD be performed. Whether oversubscription is permitted may may be specified.

- requestAllocCheck

```
virtual int requestAllocCheck(const char* planName, const
HEpochInterval& intvl, int rsId, int actId);
```

Request that an allocation be checked. This will determine if the allocation COULD be performed. This defaults to oversubscription not allowed.

- requestAllocMove

```
virtual int requestAllocMove(int actId, int sourceRsId, int
destRsId, int ovrAllowed, const char* sourcePlanName, const
char* destPlanName, const HEpochInterval& oldIntvl, const
HEpochInterval& newIntvl, int impactFlag);
```

Request that an allocation be moved. Whether oversubscription is permitted may may be specified.

- requestAllocMove

```
virtual int requestAllocMove(int actId, int sourceRsId, int
destRsId, const char* sourcePlanName, const char* destPlanName,
const HEpochInterval& oldIntvl, const HEpochInterval& newIntvl,
int impactFlag);
```

Request that an allocation be moved. The default is for oversubscription to be not allowed.

- requestAllocRqst

```
virtual int requestAllocRqst(const char* planName,
HObjCollection& reqsToSch, HObjCollection& schRqsts,
HObjCollection& notSchRqsts, const char* allocatorType);
```

Request that an allocation be performed, with constraint-checking, against a provided collection of RSchRqst(s). This is overridden from my base class to presume that oversubscription is not allowed.

- requestAllocRqst

```
virtual int requestAllocRqst(const char* planName,  
HObjCollection& reqsToSch, HObjCollection& schRqsts,  
HObjCollection& notSchRqsts, int ovrAllowed, const char*  
allocatorType);
```

Request that an allocation be performed, with constraint-checking, against a provided collection of RSchRqst(s). This is overloaded from my base class to permit the caller to specify if oversubscription is allowed or not.

- requestAlloc

```
virtual int requestAlloc(const char* planName, const  
HEpochInterval& intvl, int rsId, int actId, const char*  
allocatorType);
```

Request that an allocation be performed, with constraint-checking. This is overridden from my base class to presume that oversubscription is not allowed.

- requestAlloc

```
virtual int requestAlloc(const char* planName, const  
HEpochInterval& intvl, int rsId, int actId, int ovrAllowed,  
const char* allocatorType);
```

Request that an allocation be performed, with constraint-checking. Whether oversubscription is permitted may be specified.

- requestPlanSave

```
virtual int requestPlanSave();
```

Request to save a plan in a particular file.

- requestPlans

```
virtual int requestPlans(HObjList* plans);
```

Request all plans

- requestRsChg

```
virtual int requestRsChg(RResource* changedRs);
```

Request a resource in the pool be changed

- requestRsDel

```
virtual int requestRsDel(RResource* deletedRs);
```

Request a resource be deleted from the pool

- requestRsNew

```
virtual int requestRsNew(RResource* newRs);
```

Request a resource be added to the pool

- requestRsNms

```
virtual int requestRsNms();
```

Request to receive a list of names of all the resources in the SRM's resource pool

6.3.17 Resource_Planning_Db Class Category

6.3.17.1 Overview

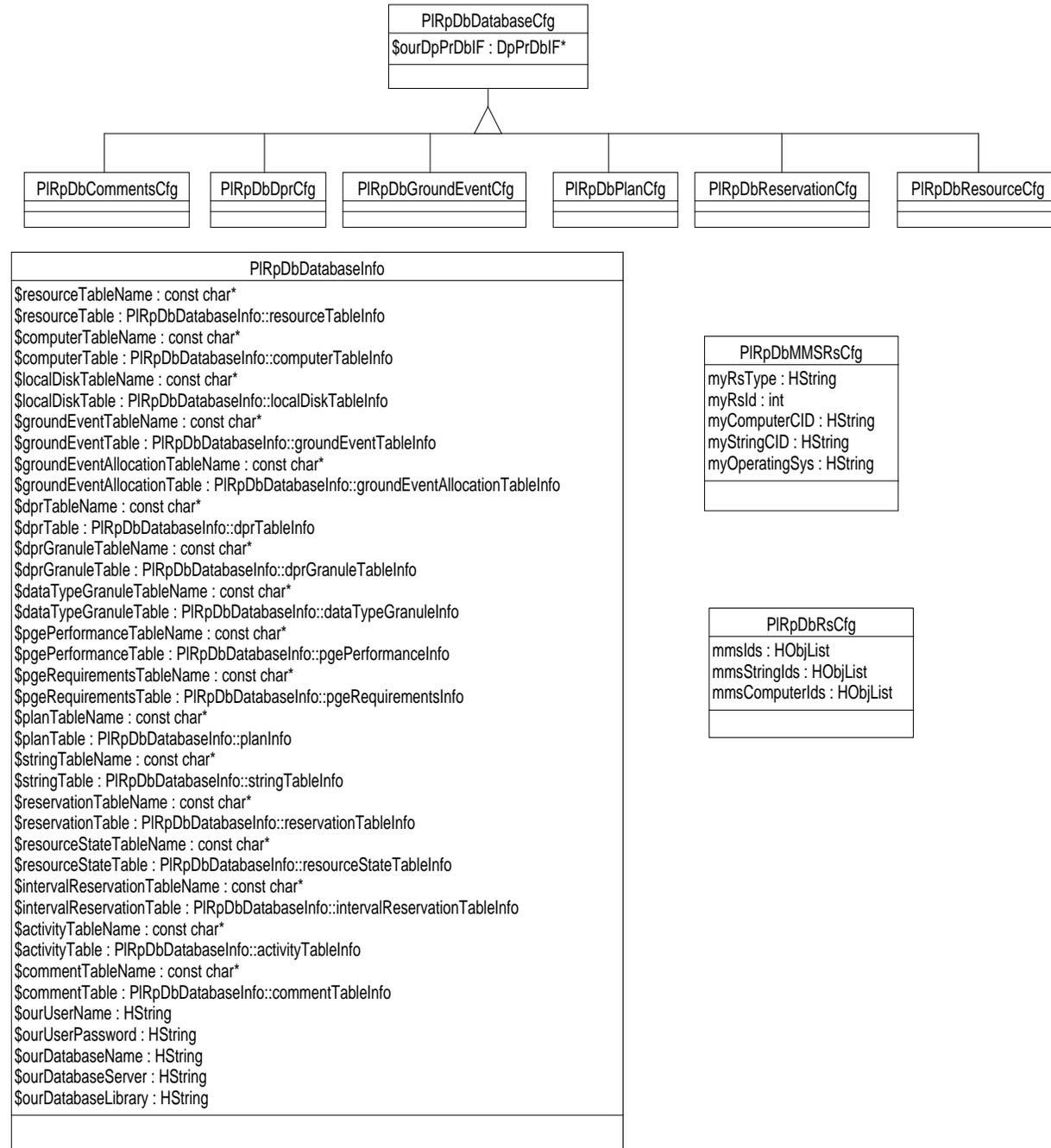


Figure 6.3.17.1-1 Resource_Planning_Db

6.3.17.2 Resource_Planning_Db Classes

6.3.17.3 PIRpDbCommentsCfg Class

Overview:

PIRpDbCommentsCfg class allows loading the comments for a resource from the database tables and saving the comment to the database tables from the GUI.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpDbDatabaseCfg

Attributes:

Constructors and Destructor:

```
PIRpDbCommentsCfg();
```

Default constructor

```
PIRpDbCommentsCfg(const PIRpDbCommentsCfg& orig);
```

Copy constructor

```
virtual ~PIRpDbCommentsCfg();
```

Destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpDbCommentsCfg );
```

PIRpDbCommentsCfg class allows loading the comments for a resource from the database tables and saving the comment to the database tables from the GUI.

- deleteComments

```
virtual int deleteComments(int rsId, HString commentType);
```

Delete existing comments

- init

```
virtual int init();
```

Initialize by connecting to the database containing resource information.

- load

```
virtual HString load(int rsId, HString commentType) const;
```

Load a string containing the comments for a given resource

- operator =

```
PlRpDbCommentsCfg& operator =(const PlRpDbCommentsCfg& orig);
```

Assignment operator

- saveComment

```
virtual int saveComment(int rsId, HString commStr, int commOrder,
HString commentType);
```

Save the comments string for a given resource

- save

```
virtual int save(int rsId, HString commentStr, HString
commentType);
```

Save the comments for a given resource

6.3.17.3.1 PIRpDbDatabaseCfg Class

Overview:

PIRpDbDatabaseCfg class serves as a base class for all HCL database interface classes.

Export Control: Public

Inheritance Relationships:

Attributes:

```
ourDpPrDbIF: DpPrDbIF*
```

PDPS Database Interface

Constructors and Destructor:

```
PlRpDbDatabaseCfg(const PlRpDbDatabaseCfg& orig);
```

Copy constructor

```
PlRpDbDatabaseCfg(const HString& userName, const HString&
userPassword, const HString& databaseName, const HString&
databaseServer, const HString& databaseLibrary);
```

Constructor takes in the user id, database password, database name, database server, and database name.

```
PlRpDbDatabaseCfg();
```

Default constructor

```
virtual ~PlRpDbDatabaseCfg();
```

Destructor

Operations:

- databaseIF

```
static DpPrDbIF* databaseIF();
```

Get my DpPrDbIF connection

- databaseIF

```
static DpPrDbIF* databaseIF(DpPrDbIF* pDpPrDbIF);
```

Set my DpPrDbIF connection

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpDbDatabaseCfg );
```

PlRpDbDatabaseCfg class serves as a base class for all HCL database interface classes.

- HDateTimeToRWDBDateTime

```
int HDateTimeToRWDBDateTime(const HDateTime& hDateTime,  
RWDBDateTime& rwDBDateTime);
```

Convert from HDateTime to RWDBDateTime and return a status

- init

```
static int init();
```

Initialize connection to database

- operator =

```
PlRpDbDatabaseCfg& operator =(const PlRpDbDatabaseCfg& orig);
```

Assignment operator

- queryAll

```
RWTValslist* queryAll(const HString& tableName) const;
```

Query the database for all rows of specified table, returning the resulting table in an array of ColValLists.

- queryByValue

```
RWTValSlist* queryByValue(const RWDBValue& value, const HString&
columnName, const HString& table) const;
```

Query table by value and return an array of resulting column/value lists.

- RWDBDateTimeToHDateTime

```
int RWDBDateTimeToHDateTime(const RWDBDateTime& rwDBDateTime,
HDateTime& hDateTime);
```

Convert from RWDBDateTime to HDateTime and return a status

6.3.17.3.2 PIRpDbDatabaseInfo Class

Overview:

PIDatabaseInfo class serves as an encapsulation of the database schema and stores database login information.

Export Control: Public

Inheritance Relationships:

Attributes:

```
activityTableName: const char*
```

Activities table name

```
activityTable: PIRpDbDatabaseInfo::activityTableInfo
```

Activities table schema

```
commentTableName: const char*
```

Comments table name

```
commentTable: PIRpDbDatabaseInfo::commentTableInfo
```

Comments table schema

```
computerTableName: const char*
```

Computer resource table name

```
computerTable: PIRpDbDatabaseInfo::computerTableInfo
```

Computer resource table schema Removed perProcessRamColumn and perProcessCpuColumn
Added ramAllocColumn

```
dataTypeGranuleTableName: const char*
```

Data type granule table name

dataTypeGranuleTable: PlRpDbDatabaseInfo::dataTypeGranuleInfo

Data type granule table schema

dprGranuleTableName: const char*

DPR/granule table name

dprGranuleTable: PlRpDbDatabaseInfo::dprGranuleTableInfo

DPR/granule table schema

dprTableName: const char*

DPR master table name

dprTable: PlRpDbDatabaseInfo::dprTableInfo

DPR master table schema

groundEventAllocationTableName: const char*

Ground event allocation table name

**groundEventAllocationTable:
PlRpDbDatabaseInfo::groundEventAllocationTableInfo**

Ground event allocation table schema

groundEventTableName: const char*

Ground event master table name

groundEventTable: PlRpDbDatabaseInfo::groundEventTableInfo

Ground event master table schema

intervalReservationTableName: const char*

Interval Reservation name

**intervalReservationTable:
PlRpDbDatabaseInfo::intervalReservationTableInfo**

Interval Reservation schema

localDiskTableName: const char*

Local disk partition resource table name

localDiskTable: PlRpDbDatabaseInfo::localDiskTableInfo

Local disk partition resource table schema Removed sysAllocationColumn and userAllocationColumn Added diskUsageColumn

ourDatabaseLibrary: HString

RW Access Library

ourDatabaseName: HString

Database name

ourDatabaseServer: HString

Database server name

ourUserName: HString

Database user Login Id

ourUserPassword: HString

Database user password

pgePerformanceTableName: const char*

PGE performance table name

pgePerformanceTable: PlRpDbDatabaseInfo::pgePerformanceInfo

PGE performance table schema

pgeRequirementsTableName: const char*

PGE resource requirements table name

pgeRequirementsTable: PlRpDbDatabaseInfo::pgeRequirementsInfo

PGE resource requirements table schema

planTableName: const char*

Plan table name

planTable: PlRpDbDatabaseInfo::planInfo

Plan table schema

reservationTableName: const char*

Reservation table name

reservationTable: PlRpDbDatabaseInfo::reservationTableInfo

Reservation table schema

resourceStateTableName: const char*

Resource state table name

resourceStateTable: PlRpDbDatabaseInfo::resourceStateTableInfo

Resource state table schema

resourceTableName: const char*

Resource master table name

```
resourceTable: PIRpDbDatabaseInfo::resourceTableInfo
```

Resource master table schmea

```
stringTableName: const char*
```

String table name

```
stringTable: PIRpDbDatabaseInfo::stringTableInfo
```

String table schema

Constructors and Destructor:

```
PIRpDbDatabaseInfo(const PIRpDbDatabaseInfo& orig);
```

Copy constructor

```
PIRpDbDatabaseInfo(const HString& userName, const HString&  
userPassword, const HString& databaseName, const HString&  
databaseServer, const HString& databaseLibrary);
```

Constructor takes in the user id, database password, database name, database server, and database name.

```
PIRpDbDatabaseInfo();
```

Default constructor

```
virtual ~PIRpDbDatabaseInfo();
```

Destructor

Operations:

- databaseLibrary

```
static int databaseLibrary(const HString& dbLibrary);
```

Set my access Library

- databaseLibrary

```
static const HString& databaseLibrary();
```

Get my access Library

- databaseName

```
static const HString& databaseName();
```

Get my database name

- databaseName

```
static int dbName(const HString& dbName);
```

Set my database name

- databaseServer

```
static const HString& databaseServer();
```

Get my server name

- databaseServer

```
static int databaseServer(const HString& dbServer);
```

Set my server name

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpDbDatabaseInfo );
```

PlRpDbDatabaseInfo class serves as an encapsulation of the database schema and stores database login information.

- operator =

```
PlRpDbDatabaseInfo& operator =(const PlRpDbDatabaseInfo& orig);
```

Assignment operator

- userName

```
static int userName(const HString& uName);
```

Set my user login id

- userName

```
static const HString& userName();
```

Get my user login id

- userPassword

```
static const HString& userPassword();
```

Get my user password

- userPassword

```
static int userPassword(const HString& uPassword);
```

Set my user password

6.3.17.3.3 PIRpDbDprCfg Class

Overview:

PIRpDbDprCfg allows loading and saving data processing request (DPR) activities (PIRpAcDpr) from and to the database tables.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpDbDatabaseCfg

Attributes:

Constructors and Destructor:

```
PIRpDbDprCfg();
```

Default constructor

```
PIRpDbDprCfg(const PIRpDbDprCfg& orig);
```

Copy constructor

```
virtual ~PIRpDbDprCfg();
```

Destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpDbDprCfg );
```

PIRpDbDprCfg allows loading and saving data processing request (DPR) activities (PIRpAcDpr) from and to the database tables.

- extractAll

```
virtual int extractAll();
```

Retrieve all ground activities from their database tables

- extractPgeInformation

```
virtual PIRpAcDpr* extractPgeInformation(PIRpAcDpr* dpr);
```

Extract PGE information for dpr from database.

- extract

```
virtual PlRpAcDpr* extract(PlRpAcDpr* dpr);
```

Extract detail information for dpr from database.

- init

```
virtual int init();
```

Initialize by connecting to the database containing DPR information.

- load

```
virtual int load();
```

Load the DPR activities from the database tables.

- makeNewObject

```
virtual HObject* makeNewObject(const DpPrDbColValList&  
thisColValList);
```

Create an instance of a DPR from a ColValList and associated detail tables.

- makeObjects

```
virtual int makeObjects(const RWTValSlist& resultsList);
```

Create new instances of DPRs from an array of ColValLists and add them to the activity pool.

- operator =

```
PlRpDbDprCfg& operator =(const PlRpDbDprCfg& orig);
```

Assignment operator

- resolveDprDependencies

```
virtual int resolveDprDependencies();
```

Create predecessor DPR lists

- resolveExternalDataDependencies

```
int resolveExternalDataDependencies();
```

Create external data dependency lists and determine earliest possible start times based on the latest arriving external data dependency.

- save

```
virtual int save();
```

Save the current DPR activity pool to the database tables.

6.3.17.3.4 PIRpDbGroundEventCfg Class

Overview:

PIRpDbGroundEventCfg allows loading and saving ground event activities (PIRpAcGroundEvent) from and to the database tables.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpDbDatabaseCfg

Attributes:

Constructors and Destructor:

```
PIRpDbGroundEventCfg();
```

Default constructor

```
PIRpDbGroundEventCfg(const PIRpDbGroundEventCfg& orig);
```

Copy constructor

```
virtual ~PIRpDbGroundEventCfg();
```

Destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpDbGroundEventCfg );
```

PIRpDbGroundEventCfg allows loading and saving ground event activities (PIRpAcGroundEvent) from and to the database tables.

- extractAll

```
virtual int extractAll();
```

Retrieve all ground event activities from their database tables

- extract

```
PIRpAcGroundEvent* extract(PIRpAcGroundEvent* newGroundEvent);
```

Extract allocations to resources for the given ground event.

- init

```
virtual int init();
```

Initialize by connecting to the database containing ground event information.

- load

```
virtual int load();
```

Load the ground event activities from the database tables.

- makeNewObject

```
virtual HObject* makeNewObject(const DpPrDbColValList&  
thisColValList);
```

Create an instance of a ground event from a ColValList and associated detail tables.

- makeObjects

```
virtual int makeObjects(const RWTValslist& resultsList);
```

Create new instances of ground events from an array of ColValLists and add them to the activity pool.

- operator =

```
PIRpDbGroundEventCfg& operator =(const PIRpDbGroundEventCfg&  
orig);
```

Assignment operator

- save

```
virtual int save();
```

Save the current ground event activity pool off to the database tables.

6.3.17.3.5 PIRpDbMMSRsCfg Class

Overview:

The PIRpDbMMSRsCfg class contains the control identification values from the configuration file received from Baseline Manager (MSS). Since no order of resources exists, the resources must be read in, loaded into a pool and then ids resolved (which computers belong to which strings, etc.)

Export Control: Public

Inheritance Relationships:

Attributes:

```
myComputerCID: HString
```

Computer Control Identifier (from MSS)

myOperatingSys: HString

Operating system for a computer

myRsId: int

Id given to resource

myRsType: HString

Type of resource (Computer, String, Disk, Hardware or Software)

myStringCID: HString

String Control Identifier (from MSS)

Constructors and Destructor:

PIRpDbMMSRsCfg();

Default constructor

~PIRpDbMMSRsCfg();

Default destructor

Operations:

- computerCID

virtual int computerCID(const char* cmptrCID);

Set computer control identifier (from MSS)

- computerCID

virtual const HString& computerCID() const;

Get computer control identifier (from MSS)

- DECLARE_LOCAL_CLASS

int DECLARE_LOCAL_CLASS(PIRpDbMMSRsCfg);

The PIRpDbMMSRsCfg class contains the control identification values from the configuration file received from Baseline Manager (MSS). Since no order of resources exists, the resources must be read in, loaded into a pool and then ids resolved (which computers belong to which strings, etc.)

- operatingSys

virtual int operatingSys(const char* theOS);

Set Operating System

- operatingSys

```
virtual const HString& operatingSys() const;
```

Get Operating System

- operator =

```
PIRpDbMMSRsCfg& operator =(const PIRpDbMMSRsCfg& cfg);
```

Assignment operator

- resourceId

```
virtual int resourceId() const;
```

Get resource id

- resourceId

```
virtual int resourceId(int rsId);
```

Set resource id

- rsType

```
virtual int rsType(const char* resourceType);
```

Set resource type

- rsType

```
virtual const HString& rsType() const;
```

Get resource type

- stringCID

```
virtual const HString& stringCID() const;
```

Get String Control Identifier (from MSS)

- stringCID

```
virtual int stringCID(const char* strCID);
```

Set String Control Identifier (from MSS)

6.3.17.3.6 PIRpDbPlanCfg Class

Overview:

PIRpDbPlanCfg allows loading and saving of Plan information (PIPIPlan) from and to the database tables.

Export Control: Public

Inheritance Relationships:

Inherits from `PIRpDbDatabaseCfg`

Attributes:

Constructors and Destructor:

```
PIRpDbPlanCfg();
```

Default constructor

```
PIRpDbPlanCfg(const PIRpDbPlanCfg& orig);
```

Copy constructor

```
virtual ~PIRpDbPlanCfg();
```

Destructor

Operations:

- `DECLARE_LOCAL_CLASS`

```
int DECLARE_LOCAL_CLASS(PIRpDbPlanCfg );
```

`PIRpDbPlanCfg` allows loading and saving of Plan information (`PIPIPlan`) from and to the database tables.

- `extractAll`

```
virtual int extractAll();
```

Retrieve all ground event activities from their database tables

- `init`

```
virtual int init();
```

Initialize by connecting to the database containing ground event information.

- `load`

```
virtual int load();
```

Load the ground event activities from the database tables.

- `makeNewObject`

```
virtual HObject* makeNewObject(const DpPrDbColValList&  
thisColValList);
```

Create an instance of a ground event from a `ColValList` and associated detail tables.

- makeObjects

```
virtual int makeObjects(const RWTValSlist& resultsList);
```

Create new instances of ground events from an array of ColValLists and add them to the activity pool.

- operator =

```
PIRpDbPlanCfg& operator =(const PIRpDbPlanCfg& orig);
```

Assignment operator

- save

```
virtual int save();
```

Save the current ground event activity pool off to the database tables.

6.3.17.3.7 PIRpDbReservationCfg Class

Overview:

PIRpDbReservationCfg allows loading and saving reservation activities (PIRpAcResourceReservation) from and to the database tables.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpDbDatabaseCfg

Attributes:

Constructors and Destructor:

```
PIRpDbReservationCfg();
```

Default constructor

```
PIRpDbReservationCfg(const PIRpDbReservationCfg& orig);
```

Copy constructor

```
virtual ~PIRpDbReservationCfg();
```

Destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpDbReservationCfg );
```

PlRpDbReservationCfg allows loading and saving reservation activities (PlRpAcResourceReservation) from and to the database tables.

- deleteReservation

```
int deleteReservation(const int id);
```

Delete reservation from the database

- dumpActPool

```
virtual int dumpActPool();
```

Dump the activity (reservation) pool

- extractAll

```
virtual int extractAll();
```

Retrieve all reservation activities from their database tables

- extractOther

```
PlRpAcResourceReservation*
```

```
extractOther(PlRpAcResourceReservation* reservation, const  
RWTValSlist& , const RWTValSlist& );
```

Fill resources and intervals associate with this reservation

- extract

```
PlRpAcResourceReservation* extract(PlRpAcResourceReservation*  
newReservation);
```

Extract allocations to resources for the given reservation.

- init

```
virtual int init();
```

Initialize by connecting to the database containing reservation information.

- load

```
virtual int load();
```

Load the reservation activities from the database tables.

- makeNewObject

```
virtual HObject* makeNewObject(const DpPrDbColValList&  
thisColValList);
```

Create an instance of a reservation from a ColValList and associated detail tables.

- makeObjects

```
virtual int makeObjects(const RWTValSlist& resultsList);
```

Create new instances of reservation from an array of ColValLists and add them to the activity pool.

- operator =

```
PIRpDbReservationCfg& operator =(const PIRpDbReservationCfg& orig);
```

Assignment operator

- saveReservation

```
int saveReservation(PIRpAcResourceReservation* reservation);
```

Insert reservation into the database

- save

```
virtual int save();
```

Save the current reservation activity pool off to the database tables.

- updateReservation

```
int updateReservation(PIRpAcResourceReservation* reservation);
```

Update reservation in the database

- updateStatus

```
int updateStatus(const int id, const char* newStatus);
```

Update state of reservation base in given reservation's id

6.3.17.3.8 PIRpDbResourceCfg Class

Overview:

PIRpDbResourceCfg class allows loading the resource pool from the database tables and saving to the database tables from the resource pool. PIRpDbResourceCfg is also responsible for creating schedulable resources and adding them to the schedulable resource pool.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpDbDatabaseCfg

Attributes:

Constructors and Destructor:

```
PIRpDbResourceCfg();
```

Default constructor

```
PIRpDbResourceCfg(const PIRpDbResourceCfg& orig);
```

Copy constructor

```
virtual ~PIRpDbResourceCfg();
```

Destructor

Operations:

- checkForDupName

```
virtual int checkForDupName(const HString& rsName);
```

Check that the resource name does not already exist

- checkReservation

```
virtual int checkReservation(PIRpRcComputer* computer);
```

Check that computer resource has not been reserved

- checkReservation

```
virtual int checkReservation(PIRpRcLocalDisk* localDisk);
```

Check that disk resource has not been reserved

- checkReservation

```
virtual int checkReservation(RResource* thisRs);
```

Check that the hardware or string resource has not been reserved

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpDbResourceCfg );
```

PIRpDbResourceCfg class allows loading the resource pool from the database tables and saving to the database tables from the resource pool. PIRpDbResourceCfg is also responsible for creating schedulable resources and adding them to the schedulable resource pool.

- deleteBaseTable

```
virtual int deleteBaseTable(int actId);
```

Delete record from base table

- deleteRsTable

```
virtual int deleteRsTable(const HString& tableName, const  
HString& columnName, int rsId);
```

Delete record from appropriate resource table

- deleteRs

```
virtual int deleteRs(RResource* rs);
```

Delete the selected resource from the database tables.

- dumpRsPool

```
virtual int dumpRsPool();
```

Dump the resource pool

- extractAll

```
virtual int extractAll();
```

Retrieve all resource objects from their database tables

- extract

```
virtual PlRpRcResource* extract(PlRpRcComputer* newComputer);
```

Overloaded extraction methods for completing the construction of computer resource objects from the database.

- extract

```
virtual PlRpRcResource* extract(PlRpRcLocalDisk* newLocalDisk);
```

Overloaded extraction methods for completing the construction of local disk resource objects from the database.

- handleConnections

```
virtual int handleConnections();
```

Resolve connections between associated resources.

- handleConnections

```
virtual int handleConnections(int cpuFlag);
```

Resolve connections between associated resources.

- init

```
virtual int init();
```

Initialize by connecting to the database containing resource information.

- insertBaseTable

```
virtual int insertBaseTable(const HString& tableName, RResource*  
rs, const HString& rsType, int actId);
```

Insert record into base table

- insertComputerTable

```
virtual int insertComputerTable(const HString& tableName,  
PlRpRcComputer* computer);
```

Insert record into computer resource table

- insertLocalDiskTable

```
virtual int insertLocalDiskTable(const HString& tableName,  
PlRpRcLocalDisk* localDisk);
```

Insert record into local disk resource table

- insertStringTable

```
virtual int insertStringTable(const HString& tableName,  
RResource* rs);
```

Insert record into string table

- load

```
virtual int load(const HString& tag);
```

Load the resource pool from the file from MSS

- load

```
virtual int load();
```

Load the resource pool from the database tables.

- makeNewObject

```
virtual HObject* makeNewObject(const DpPrDbColValList&  
thisColValList);
```

Create an instance of a resource from a ColValList and associated detail tables.

- makeObjects

```
virtual int makeObjects(const RWTValSlist resultsList);
```

Create new instances of resources from an array of ColValLists and add them to the resource pool.

- operator =

```
PlRpDbResourceCfg& operator =(const PlRpDbResourceCfg& orig);
```

Assignment operator

- removeComputers

```
virtual int removeComputers(int rsId);
```

Remove associated computers from a deleted string
- removeDisks

```
virtual int removeDisks(int rsId);
```

Remove associated disks from a deleted computer
- saveComputer

```
virtual int saveComputer(PlRpRcComputer* computerRs, int actId);
```

Save a computer resource to the database.
- saveDevice

```
virtual int saveDevice(PlRpRcResource* deviceRs, int actId);
```

Save hardware device resource to the database.
- saveLocalDisk

```
virtual int saveLocalDisk(PlRpRcLocalDisk* diskRs, int actId);
```

Save a local disk resource to the database.
- saveString

```
virtual int saveString(RResource* rs, int actId);
```

Save a string resource to the database.
- save

```
virtual int save(RResource* rs, int activityId, int fromMSSFlag);
```

Save the current resource to the database tables.
- save

```
virtual int save();
```

Save the current resource pool off to the database tables.
- save

```
virtual int save(int mssFlag);
```

Save the current resource pool off to the database tables.
- setTableStatus

```
virtual int setTableStatus(int lockStatus, int rsId);
```

Lock or unlock resource tables (for Resource Management)

- updateBaseTable

```
virtual int updateBaseTable(const HString& tableName, RResource*  
rs, const HString& rsType, int actId);
```

Update record into base table

- updateComputerIds

```
virtual int updateComputerIds(PlRpRcComputer* computer);
```

Update computer ids for local disks

- updateComputerTable

```
virtual int updateComputerTable(const HString& tableName,  
PlRpRcComputer* computer);
```

Update record into computer table

- updateComputer

```
virtual int updateComputer(PlRpRcComputer* computerRs, int  
actId);
```

Update computer resource to the database.

- updateDevice

```
virtual int updateDevice(PlRpRcResource* deviceRs, int actId);
```

Update hardware device resource to the database.

- updateLocalDiskTable

```
virtual int updateLocalDiskTable(const HString& tableName,  
PlRpRcLocalDisk* localDisk);
```

Update record into local disk table

- updateLocalDisk

```
virtual int updateLocalDisk(PlRpRcLocalDisk* diskRs, int actId);
```

Update disk resource to the database.

- updateStringIds

```
virtual int updateStringIds(PlRpRcString* strRs);
```

Update string ids for computers

- updateStringTable

```
virtual int updateStringTable(const HString& tableName,  
PlRpRcString* strRs);
```

Update record into string table

- updateString

```
virtual int updateString(PlRpRcString* strRs, int actId);
```

Update string resource to the database.

- update

```
virtual int update(RResource* rs, int activityId);
```

Update the current resource to the database tables.

6.3.17.3.9 PIRpDbRsCfg Class

Overview:

The PIRpDbRsCfg class is the interface between Resource Planning and Baseline Manager (MSS). This class loads the resource pools from the file received from Baseline Manager and then updates the database tables with this information.

Export Control: Public

Inheritance Relationships:

Attributes:

```
mmsComputerIds: HObjList
```

List of computer resources from MSS

```
mmsIds: HObjList
```

List of resources (all types) from MSS

```
mmsStringIds: HObjList
```

List of string resources from MSS

Constructors and Destructor:

```
PlRpDbRsCfg();
```

Default constructor

```
virtual ~PlRpDbRsCfg();
```

Default destructor

Operations:

- close

```
virtual void close();
```

Close the file stream

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpDbRsCfg );
```

The PlRpDbRsCfg class is the interface between Resource Planning and Baseline Manager (MSS). This class loads the resource pools from the file received from Baseline Manager and then updates the database tables with this information.

- handleComputer

```
virtual int handleComputer();
```

Handle loading of Computers

- handleDevice

```
virtual int handleDevice();
```

Handle loading of Hardware devices

- handleKeyword

```
virtual int handleKeyword(const HString& kywd);
```

Handle keyword

- handleLocalDisk

```
virtual int handleLocalDisk();
```

Handle loading of Disk partitions

- handleOS

```
virtual int handleOS();
```

Handle loading of operating system

- handleString

```
virtual int handleString();
```

Handle loading of Strings

- operator =

```
PlRpDbRsCfg& operator =(const PlRpDbRsCfg& cfg);
```

Assignment operator

- position

```
virtual int position();
```

Handle positioning

- process

```
virtual int process();
```

Handle processing file

- resolveComputerIds

```
virtual int resolveComputerIds();
```

Iterate through the list of computers, disks and operating sys and match up the disks and operating sys to the appropriate computer (based on control identifier from MSS)

- resolveStringIds

```
virtual int resolveStringIds();
```

Iterate through the list of strings and computers and match up the computers to the appropriate string (based on control identifier from MSS)

- setTag

```
virtual void setTag(const char* name);
```

Set tag name

- trimBlanks

```
virtual HString trimBlanks(HString strValue);
```

Trim trailing blanks from strings read in from configuration file

6.3.18 Resource_Planning_Di Class Category

6.3.18.1 Overview

PIRpDiAppl
myWindow : DWindow* myFallbacks : HObjList myFallbackArray : char**
DECLARE_ABS_CLASS() PIRpDiAppl() PIRpDiAppl() makeWindow() ~PIRpDiAppl() create() setFallbackResources() setNotifier() run() window() closeAppl() quit() connectionMade() \$winManagerClose() operator =() displayWinAtRun() catchCloseEvent() cleanupForWinDestroy()

6.3.18.2 Resource_Planning_Di

6.3.18.3 PIRpDiAppl

Overview:

class PIRpDiAppl (ECS Motif Application Class)

An instance of this class is a type of resource model client display application. PIRpDiAppl includes behavior common to all ECS resource model client display applications.

NOTE: PIRpDiAppl copies the functionality of DAppl in the md and mdisp libraries but dif the create by allowing the shell to resize, and in the run by not calling displayW after a connection to the resource model has been made

Export Control: Public

Inheritance Relationships:

Inherits from `PIRpCIAppl`

Attributes:

myFallbackArray: char**

My Xm resource fallback array

myFallbacks: HObjList

Collection of HStrings with my fallback resources.

myWindow: DWindow*

My application window

Constructors and Destructor:

PIRpDiAppl();

Default constructor

PIRpDiAppl(const PIRpDiAppl&);

Copy constructor (Disallows copying)

virtual ~PIRpDiAppl();

Destructor

Operations:

- `catchCloseEvent`

virtual void catchCloseEvent();

Called by `PIRpDiAppl::run` to catch the window Close event.

- `cleanupForWinDestroy`

virtual void cleanupForWinDestroy();

Called by `closeAppl` just before top-level window is destroyed. Override if you want to do something before this happens.

- `closeAppl`

virtual void closeAppl();

Call from the window if you would like to quit the application.

- `connectionMade`

```
virtual int connectionMade();
```

This member is called when the srm noid has made a connection to the resource model. Overrides PIRpCIAppl::connectionMade() to call displayWinAtRun() to realize the window

- create

```
virtual int create(const char* appClass, int& argc, char** argv, XrmOptionDescList options, Cardinal numOptions);
```

This create calls makeWindow, creates a shell, and calls myWindow->createDisplay. If you want to override this functionality, you will still need to pass myWindow::create(..) a shell widget as shown in this function. NOTE: this is the only function (so far) that diverges from the member functions in DAppl in that it allows the shell to resize.

- DECLARE_ABS_CLASS

```
int DECLARE_ABS_CLASS(PIRpDiAppl );
```

class PIRpDiAppl (ECS Motif Application Class)

An instance of this class is a type of resource model client display application. PIRpDiAppl includes behavior common to all ECS resource model client display applications.

NOTE: PIRpDiAppl copies the functionality of DAppl in the md and mdisp libraries but differs in the create by allowing the shell to resize, and in the run by not calling displayWinAtRun() until after a connection to the resource model has been made

- displayWinAtRun

```
virtual void displayWinAtRun();
```

Display the window at run. default behavior is to realize it

- makeWindow

```
virtual DWindow* makeWindow() = 0;
```

Override this class in your derived class to make the window (i.e., return new DerivedWindow();).

- operator =

```
PIRpDiAppl& operator =(const PIRpDiAppl& );
```

Assignment operator (Disallows assignments)

- quit

```
virtual void quit();
```

Quit the application. Overrides PIRpCIAppl::quit to close the display application

- run

```
virtual void run();
```

Run the application. Create(..) should have already been called.

- setFallbackResources

```
virtual int setFallbackResources(char** fallbackList);
```

List of char** arguments with fallback resources. This can be called anytime prior to PIRpDiAppl::create().

```
+ an example of use: ++ static char *fallbacks[] = { + "hmi.mainWin.background: papaya  
whip", + "hmi.mainWin.foreground: pale violet red", + "hmi.mainWin.btn.labelString: Push  
Me, Dude", + "*fontList: *-palatino*-r*--14-*", + NULL}; /* don't forget this! */ ++  
myAppl.setFallbackResources( fallbacks);
```

- setNotifier

```
virtual int setNotifier(HNotifier* nf);
```

Set the notifier for this class

- window

```
virtual DWindow* window();
```

Get the window data.

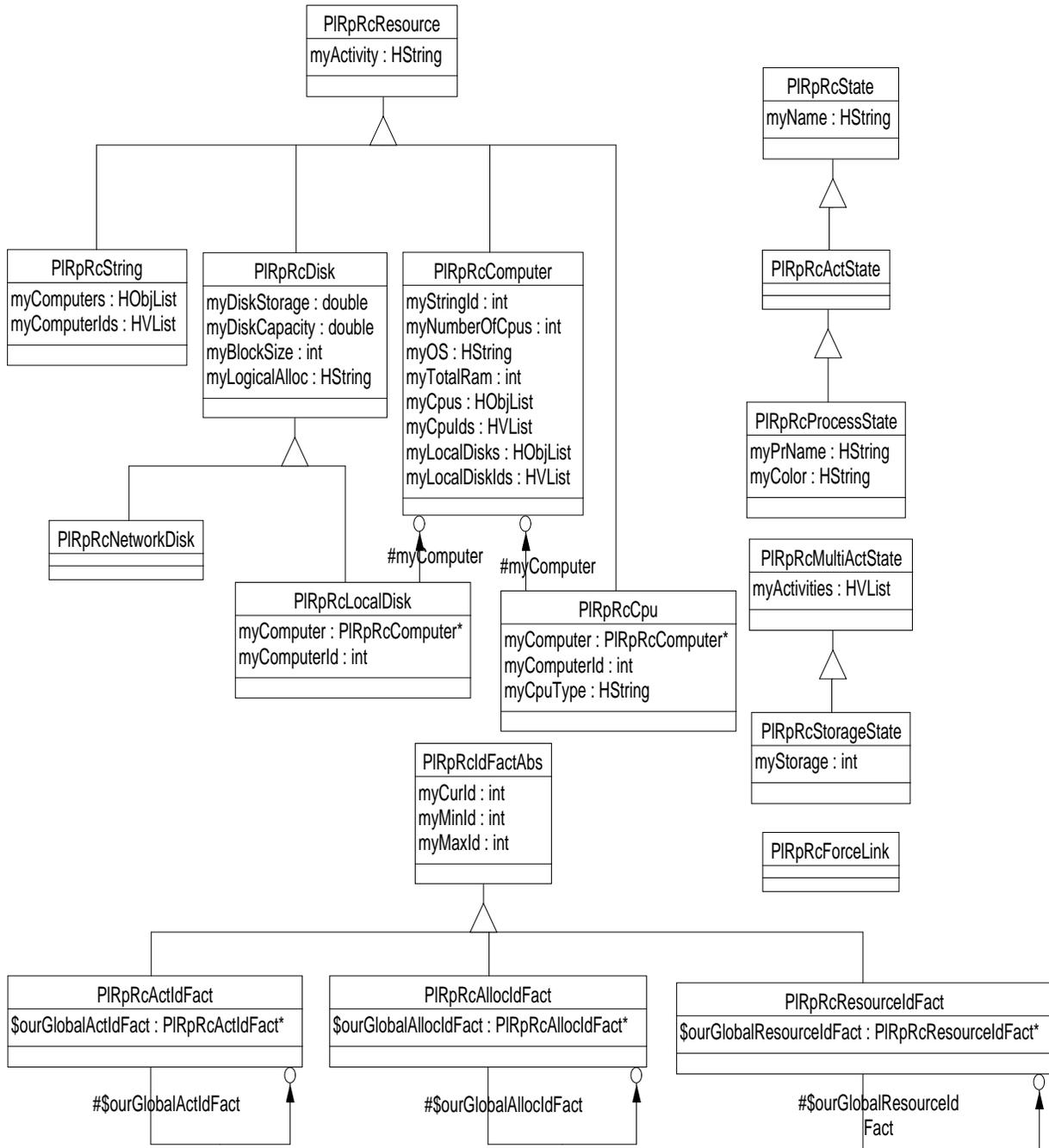
- winManagerClose

```
static void winManagerClose(Widget protocolWidget, XtPointer  
clientData, XtPointer callData);
```

The function called on window manager menu "close." Cleans up the application and exits.

6.3.19 Resource_Planning_Rc Class Category

6.3.19.1 Overview



6.3.19.2 Resource_Planning_Rc Classes

6.3.19.3 PIRpRcActIdFact Class

Overview:

class PIRpRcActIdFact (Unique activity id factory)

Instances of PIRpRcActIdFact are activity ID generators. You can ask them for the next allowable activity ID that can be assigned. This class initializes its activity ID range from the current activity pool for now. Max ID is presumed to be MAXINT.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcIdFactAbs

Attributes:

`ourGlobalActIdFact: PIRpRcActIdFact*`

Current global ID.

Constructors and Destructor:

`PIRpRcActIdFact();`

Default constructor

`PIRpRcActIdFact(const PIRpRcActIdFact& orig);`

Copy constructor, restricted copy

`virtual ~PIRpRcActIdFact();`

Destructor

Operations:

- DECLARE_CLASS

`int DECLARE_CLASS(PIRpRcActIdFact);`

class PIRpRcActIdFact (Unique activity id factory)

Instances of PIRpRcActIdFact are activity ID generators. You can ask them for the next allowable activity ID that can be assigned. This class initializes its activity ID range from the current activity pool for now. Max ID is presumed to be MAXINT.

- init

`virtual int init();`

Initialize. Set myCurId, myMinId, and myMaxId to some initial values, based upon the contents of the current activity pool. Max ID value is presumed to be MAXINT.

- nextActId

```
static int nextActId();
```

Return the next available ID in the global PIRpRcActIdFact.

- operator =

```
PIRpRcActIdFact& operator =(const PIRpRcActIdFact& orig);
```

Assignment operator, restricted assignment

- theActIdFact

```
static PIRpRcActIdFact* theActIdFact();
```

Get the global PIRpRcActIdFact.

- theActIdFact

```
static int theActIdFact(PIRpRcActIdFact* idFactory);
```

Set the global PIRpRcActIdFact. This class does not assume ownership.

6.3.19.3.1 PIRpRcActState Class

Overview:

class PIRpRcActState (Activity Resource State)

Instances of PIRpRcActState model the state of a resource during which that resource is in the process of executing an activity.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcState

Attributes:

Constructors and Destructor:

```
PIRpRcActState(const HEpochInterval& intvl, int actId, const char* name);
```

Constructor with interval, activity id and name specified

```
PIRpRcActState(const PIRpRcActState& orig);
```

Copy constructor

```
PIRpRcActState(const HEpochInterval& intvl, int actId);
```

Constructor with interval and activity id

```
PIRpRcActState();
```

Default constructor

```
virtual ~PIRpRcActState();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpRcActState );
```

class PIRpRcActState (Activity Resource State)

Instances of PIRpRcActState model the state of a resource during which that resource is in the process of executing an activity.

- operator =

```
PIRpRcActState& operator =(const PIRpRcActState& orig);
```

Assignment operator

6.3.19.3.2 PIRpRcAllocIdFact Class

Overview:

class PIRpRcAllocIdFact (Unique allocation id factory)

Instances of PIRpRcAllocIdFact are allocation ID generators. You can ask them for the next allowable allocation ID that can be assigned. This class initializes its allocation ID range from the current allocations associated with activities in the activity pool for now. Max ID is presumed to be MAXINT.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcIdFactAbs

Attributes:

```
ourGlobalAllocIdFact: PIRpRcAllocIdFact*
```

Current global ID.

Constructors and Destructor:

```
PIRpRcAllocIdFact();
```

Default constructor

```
PIRpRcAllocIdFact(const PIRpRcAllocIdFact& orig);
```

Copy constructor, restricted copy

```
virtual ~PIRpRcAllocIdFact();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpRcAllocIdFact );
```

class PIRpRcAllocIdFact (Unique allocation id factory)

Instances of PIRpRcAllocIdFact are allocation ID generators. You can ask them for the next allowable allocation ID that can be assigned. This class initializes its allocation ID range from the current allocations associated with activities in the activity pool for now. Max ID is presumed to be MAXINT.

- init

```
virtual int init();
```

Initialize. Set myCurId, myMinId, and myMaxId to some initial values, based upon the contents of the current activity allocations in the activity pool Max ID value is presumed to be MAXINT.

- nextAllocId

```
static int nextAllocId();
```

Return the next available ID in the global PIRpRcAllocIdFact.

- operator =

```
PIRpRcAllocIdFact& operator =(const PIRpRcAllocIdFact& orig);
```

Assignment operator, restricted assignment

- theAllocIdFact

```
static PIRpRcAllocIdFact* theAllocIdFact();
```

Get the global PIRpRcAllocIdFact.

- theAllocIdFact

```
static int theAllocIdFact(PIRpRcAllocIdFact* idFactory);
```

Set the global PIRpRcAllocIdFact. This class does not assume ownership.

6.3.19.3.3 PIRpRcComputer Class

Overview:

Instances of PIRpRcComputer are multi-processors with a known configuration of associated disks.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcResource

Attributes:

myCpuIds: HVList

My list of CPU ids corresponding to the above list of CPUs. I may have ids without having CPUs if I have just been loaded from a stream (see resolveIds() below).

myCpus: HObjList

List of associated CPUs with this computer; I own these CPUs exclusively

myLocalDiskIds: HVList

My list of Local disk ids corresponding to the above list of LocalDisks. I may have ids without having disks if I have just been loaded from a stream (see resolveIds() below).

myLocalDisks: HObjList

List of LocalDisks associated with this computer; I own these disks exclusively and am in charge of deleting them

myNumberOfCpus: int

Number of CPUs in this computer

myOS: HString

Operating system for this computer

myStringId: int

String id associated with this computer

myTotalRam: int

Total ram for this computer

Constructors and Destructor:

PIRpRcComputer();

Default constructor

```
PlRpRcComputer(const PlRpRcComputer& orig);
```

copy constructor

```
virtual ~PlRpRcComputer();
```

Destructor

Operations:

- addAssoc

```
virtual int addAssoc(HObject* rs);
```

Tell me that the given resource is owned by this computer.

- addCpu

```
virtual int addCpu(RResource* cpu);
```

Add a CPU to this computer.

- addDisk

```
virtual int addDisk(RResource* disk);
```

Add a disk to this computer.

- copyOver

```
virtual int copyOver(const PlRpRcComputer& orig);
```

Copy over this computer resource

- copyOver

```
virtual int copyOver(const RResource& orig);
```

Copy over this resource

- cpus

```
virtual HObjIter* cpus();
```

Return an iterator over this computers CPU list. The consumer must delete the iterator when done.

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpRcComputer );
```

Instances of PlRpRcComputer are multi-processors with a known configuration of associated disks.

- degrade

```
virtual int degrade(const HEpochInterval& intvl, const char* listName);
```

Create an instance of RDegState and add to the specified list or the degradation list if no list name is present. If the state was successfully created and added, TRUE will be returned. Otherwise FALSE will be returned. Overrides base class to propagate degradation to this computers cpus

- degrade

```
virtual int degrade(RDegState* state, const char* listName);
```

Use the given state to degrade me on the given list. If the state was successfully added, TRUE will be returned. Otherwise FALSE will be returned. Overrides base class to propagate degradation to this computers cpus

- get

```
int get(istream& );
```

Get computer resource from stream

- localDisks

```
virtual HObjIter* localDisks();
```

Return an iterator over this computers LocalDisk list. The consumer must delete the iterator when done.

- numberOfCpus

```
virtual int numberOfCpus(int nCpus);
```

Set number of CPUs for this computer

- numberOfCpus

```
virtual int numberOfCpus() const;
```

Get number of CPUs for this computer

- operatingSys

```
virtual int operatingSys(const char* os);
```

Set operating system for this computer

- operatingSys

```
virtual const HString& operatingSys() const;
```

Get operating system for this computer

- operator =

```
PLRpRcComputer& operator =(const PLRpRcComputer& orig);
```

Assignment operator

- put

```
int put(ostream& ) const;
```

Put computer resource onto stream

- remDegrade

```
virtual int remDegrade(const HEpochInterval& intvl, const char* listName);
```

Remove all degradation states that overlap the given time interval in the given list or the degradation list if no list name is present. Overrides base class to propagate degradation to this computers cpus

- removeCpu

```
virtual int removeCpu(RResource* cpu);
```

Remove a CPU from this computer.

- removeDisk

```
virtual int removeDisk(RResource* disk);
```

Remove a disk from this computer.

- resolveCpuIds

```
virtual void resolveCpuIds();
```

Try to resolve any CPU ids to CPUs from the resource pool. CPUs are passed around (ipc, to/from streams) using their reference resource id number. I may know a bunch of ids without yet having pointers to resources. This behavior will resolve any unknown ids by attempting to look them up in the RResourcePool.

- resolveLocalDiskIds

```
virtual void resolveLocalDiskIds();
```

Try to resolve Local Disk ids to disks in the same manner as for CPUs above.

- stringId

```
virtual int stringId(int strId);
```

Set string Id for this computer

- stringId

```
virtual int stringId() const;
```

Get string Id for this computer

- totalRam

```
virtual int totalRam(int tRam);
```

Set total ram this computer

- totalRam

```
virtual int totalRam() const;
```

Get total ram this computer

- xdr

```
int xdr(XDR* );
```

Get/put computer resource onto an xdr stream

6.3.19.3.4 PIRpRcCpu Class

Overview:

Instances of PIRpRcCpu are processors associated with a particular computer and are of a certain type (Sun, SGI, etc.)

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcResource

Attributes:

```
myComputerId: int
```

My computer id that I am a part of

```
myComputer: PIRpRcComputer*
```

My computer that I am a part of

```
myCpuType: HString
```

My cpu type

Constructors and Destructor:

```
PIRpRcCpu();
```

Default constructor

```
PIRpRcCpu(const PIRpRcCpu& orig);
```

Copy constructor

```
virtual ~PIRpRcCpu();
```

Destructor

Operations:

- computerId

```
virtual void computerId(const int id);
```

Set my associated computer id

- computerId

```
virtual int computerId();
```

Get my associated computer id

- computer

```
virtual int computer(PlRpRcComputer* computer);
```

Set my associated computer name

- computer

```
virtual PlRpRcComputer* computer();
```

Get my associated computer name

- cpuType

```
virtual const HString& cpuType() const;
```

Get my associated cpu type

- cpuType

```
virtual void cpuType(const HString& computerType);
```

Set my associated cpu type

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpRcCpu );
```

Instances of PlRpRcCpu are processors associated with a particular computer and are of a certain type (Sun, SGI, etc.)

- get

```
int get(istream& );
```

Get CPU object from a stream

- operator =

```
PlRpRcCpu& operator =(const PlRpRcCpu& orig);
```

Assignment operator

- put

```
int put(ostream& ) const;
```

Put CPU object onto a stream

- rsStateType

```
virtual HClassDesc& rsStateType();
```

Return the class description of my type of resource state.

- xdr

```
int xdr(XDR* );
```

Set/get CPU object from a XDR stream

6.3.19.3.5 PIRpRcDisk Class

Overview:

Instances of PIRpRcDisk are computer resources capable of providing disk storage and disk capacity information.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcResource

Attributes:

```
myBlockSize: int
```

Block size for this disk

```
myDiskCapacity: double
```

Amount of total storage available on this disk

```
myDiskStorage: double
```

Amount of current storage used on this disk

```
myLogicalAlloc: HString
```

Logical allocation (either "system" or "user")

Constructors and Destructor:

```
PIRpRcDisk();
```

Default constructor

```
PIRpRcDisk(const PIRpRcDisk& orig);
```

Copy constructor

```
virtual ~PlRpRcDisk();
```

Destructor

Operations:

- blockSize

```
virtual int blockSize(int blSize);
```

Set the block size

- blockSize

```
virtual int blockSize() const;
```

Get the block size

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpRcDisk );
```

Instances of PlRpRcDisk are computer resources capable of providing disk storage and disk capacity information.

- diskCapacity

```
virtual double diskCapacity() const;
```

Get the disk capacity

- diskCapacity

```
virtual void diskCapacity(const double diskCapacity);
```

Set the disk capacity

- diskStorage

```
virtual int diskStorage() const;
```

Get the current disk storage

- diskStorage

```
virtual void diskStorage(const double diskStorage);
```

Set the current disk storage

- get

```
int get(istream& );
```

Get disk resource from stream

- logicalAlloc

```
virtual const HString& logicalAlloc() const;
```

Get the logical allocation (either "system" or "user")

- logicalAlloc

```
virtual int logicalAlloc(const char* logAll);
```

Set the logical allocation (either "system" or "user")

- operator =

```
PIRpRcDisk& operator =(const PIRpRcDisk& orig);
```

Assignment operator

- put

```
int put(ostream& ) const;
```

Put disk resource to stream

- rsStateType

```
virtual HClassDesc& rsStateType();
```

Return the HClassDesc of the resource states belonging to a PIRpRcDisk

- xdr

```
int xdr(XDR* );
```

Get/put disk resource onto a stream

6.3.19.3.6 PIRpRcForceLink Class

Overview:

class PIRpRcForceLink (force link for ECS resource model classes) Instances of PIRpRcForceLink are used to force linkage of ECS resource model classes which may not be present at link time.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PIRpRcForceLink();
```

Default constructor

```
PIRpRcForceLink(const PIRpRcForceLink& orig);
```

Copy constructor

```
virtual ~PIRpRcForceLink();
```

Destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpRcForceLink );
```

class PIRpRcForceLink (force link for ECS resource model classes) Instances of PIRpRcForceLink are used to force linkage of ECS resource model classes which may not be present at link time.

- linkage

```
void linkage();
```

Links designated classes at link time

- operator =

```
PIRpRcForceLink& operator =(const PIRpRcForceLink& orig);
```

Assignment operator

6.3.19.3.7 PIRpRcIdFactAbs Class

Overview:

class PIRpRcIdFactAbs (Unique abstract id factory) Instances of PIRpRcIdFactAbs are abstract base class id generators. You can ask them for the next allowable id that can be assigned. This class has the notion of a high and low limits for ids that an id factory can return

Export Control: Public

Inheritance Relationships:

Attributes:

```
myCurId: int
```

My id to be assigned next.

```
myMaxId: int
```

The maximum id that can be assigned.

```
myMinId: int
```

The minimum id that can be assigned.

Constructors and Destructor:

```
PlRpRcIdFactAbs();
```

Default constructor

```
PlRpRcIdFactAbs(const PlRpRcIdFactAbs& orig);
```

Copy constructor, restricted

```
virtual ~PlRpRcIdFactAbs();
```

Destructor

Operations:

- DECLARE_ABS_CLASS

```
int DECLARE_ABS_CLASS(PlRpRcIdFactAbs );
```

class PlRpRcIdFactAbs (Unique abstract id factory) Instances of PlRpRcIdFactAbs are abstract base class id generators. You can ask them for the next allowable id that can be assigned. This class has the notion of a high and low limits for ids that an id factory can return

- get

```
int get(istream& istr);
```

Get myself from a stream

- init

```
virtual int init() = 0;
```

Pure virtual initialize. Set myCurId, myMinId, and myMaxId to some initial values. Derived classes must override to init themselves according to other heuristics.

- maxId

```
virtual int maxId() const;
```

Get for maximum ID.

- maxId

```
virtual int maxId(int id);
```

Set for maximum ID.

- minId

```
virtual int minId() const;
```

Get for minimum ID.

- minId

```
virtual int minId(int id);
```

Set for minimum ID.

- nextIdLocal

```
virtual int nextIdLocal();
```

Return the next available ID. If we run off the end of our legal range, then return -1.

- operator =

```
PIRpRcIdFactAbs& operator =(const PIRpRcIdFactAbs& orig);
```

Assignment operator, restricted

- put

```
int put(ostream& ostr) const;
```

Put myself onto a stream

- xdr

```
int xdr(XDR* xdrs);
```

Get/put myself onto an xdr stream

6.3.19.3.8 PIRpRcLocalDisk Class

Overview:

Instances of PIRpRcLocalDisk are Disk storage units which are "Locally" or directly attached to a Computer and are therefore owned by that computer.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcDisk

Attributes:

```
myComputerId: int
```

The id of the computer of which this local disk is a part

`myComputer: PlRpRcComputer*`

The computer of which this local disk is a part, and its id (for visibility)

Constructors and Destructor:

`PlRpRcLocalDisk();`

Default constructor

`PlRpRcLocalDisk(const PlRpRcLocalDisk& orig);`

Copy constructor

`virtual ~PlRpRcLocalDisk();`

Destructor

Operations:

- computerId

`virtual void computerId(const int id);`

Set my associated computer id

- computerId

`virtual int computerId();`

Get my associated computer id

- computer

`virtual PlRpRcComputer* computer();`

Get my associated computer name

- computer

`virtual int computer(PlRpRcComputer* computer);`

Set my associated computer name

- DECLARE_CLASS

`int DECLARE_CLASS(PlRpRcLocalDisk);`

Instances of PlRpRcLocalDisk are Disk storage units which are "Locally" or directly attached to a Computer and are therefore owned by that computer.

- get

`int get(istream&);`

Get local disk resource from stream

- operator =

```
PlRpRcLocalDisk& operator =(const PlRpRcLocalDisk& orig);
```

Assignment operator

- put

```
int put(ostream& ) const;
```

Put local disk resource to stream

- xdr

```
int xdr(XDR* );
```

Get/put local disk resource onto a stream

6.3.19.3.9 PlRpRcMultiActState Class

Overview:

class PlRpRcMultiActState (resource state for multiple activities)

This class is a RRsState that can keep track of associations between it and multiple RActivity 'tasking' requests.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myActivities: HVList
```

List of my activities

Constructors and Destructor:

```
PlRpRcMultiActState(const HEpochInterval& orig);
```

Constructor with specific time interval

```
PlRpRcMultiActState(const PlRpRcMultiActState& orig);
```

Copy constructor

```
PlRpRcMultiActState();
```

Default constructor

```
virtual ~PlRpRcMultiActState();
```

Destructor

Operations:

- activities

```
virtual HVIter* activities();
```

Access to my allocation activities Consumer responsible for deleting iter.

- addId

```
virtual int addId(int id);
```

Add an activity allocation, by activity id

- anyId

```
virtual int anyId() const;
```

Returns TRUE/FALSE if there are any activity ids within me

- containsId

```
int containsId(int id);
```

Returns TRUE/FALSE if this activity id is contained within me.

- copyOver

```
virtual int copyOver(const PlRpRcMultiActState& );
```

Create new copies of allocation ids in the source list

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpRcMultiActState );
```

class PlRpRcMultiActState (resource state for multiple activities)

This class is a RRsState that can keep track of associations between it and multiple RActivity 'tasking' requests.

- get

```
virtual int get(istream& istr);
```

Get my contents from a stream.

- operator =

```
PlRpRcMultiActState& operator =(const PlRpRcMultiActState&  
orig);
```

Assignment operator

- put

```
virtual int put(ostream& ostr) const;
```

Put my contents onto a stream.

- removeId

```
virtual int removeId(int id);
```

Remove an activity allocation, by activity id

- xdr

```
virtual int xdr(XDR* xdrs);
```

ENCODE/DECODE my contents to/from an xdr stream.

6.3.19.3.10 PIRpRcNetworkDisk Class

Overview:

Instances of PIRpRcNetworkDisk are Disk storage units which are "Network Attached" and behave like NFS mounted disks. They are not owned by another RResource.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcDisk

Attributes:

Constructors and Destructor:

```
PIRpRcNetworkDisk();
```

Default constructor

```
PIRpRcNetworkDisk(const PIRpRcNetworkDisk& orig);
```

Copy constructor

```
virtual ~PIRpRcNetworkDisk();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpRcNetworkDisk );
```

Instances of PIRpRcNetworkDisk are Disk storage units which are "Network Attached" and behave like NFS mounted disks. They are not owned by another RResource.

- get

```
int get(istream& );
```

Get network disk object from a stream

- operator =

```
PIRpRcNetworkDisk& operator =(const PIRpRcNetworkDisk& orig);
```

Assignment operator

- put

```
int put(ostream& ) const;
```

Put network disk object onto a stream

- xdr

```
int xdr(XDR* );
```

Set/get network disk object from a XDR stream

6.3.19.3.11 PIRpRcProcessState Class

Overview:

Instances of PIRpRcProcessState are classes that represent the current processing state of the CPU.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcActState

Attributes:

```
myColor: HString
```

My color

```
myPrName: HString
```

My production request name

Constructors and Destructor:

```
PIRpRcProcessState(const PIRpRcProcessState& orig);
```

Copy constructor

```
PlRpRcProcessState(const HEpochInterval& , int actId);
```

Constructor with interval specified

```
PlRpRcProcessState(const HEpochInterval& intvl, int actId, const char* name);
```

Constructor with interval, activity id and name specified

```
PlRpRcProcessState();
```

Default constructor

```
virtual ~PlRpRcProcessState();
```

Destructor

Operations:

- color

```
virtual const HString& color() const;
```

Get my color

- color

```
virtual void color(const char* );
```

Set my color

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpRcProcessState );
```

Instances of PlRpRcProcessState are classes that represent the current processing state of the CPU.

- get

```
int get(istream& );
```

Get myself from a stream

- operator =

```
PlRpRcProcessState& operator =(const PlRpRcProcessState& orig);
```

Assignment operator

- prName

```
virtual const HString& prName() const;
```

Get the name of production request

- prName

```
virtual void prName(const char* );
```

Set the name of production request

- put

```
int put(ostream& ) const;
```

Put myself onto a stream

- xdr

```
int xdr(XDR* );
```

Set/get myself from a XDR stream

6.3.19.3.12 PIRpRcResourceIdFact Class

Overview:

Instances of PIRpRcResourceIdFact are unique activity ID generators. You can ask them for the next allowable activity ID that can be assigned. This class initializes its activity ID range from the current activity pool for now. Max ID is presumed to be MAXINT.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcIdFactAbs

Attributes:

```
ourGlobalResourceIdFact: PIRpRcResourceIdFact*
```

Current global ID.

Constructors and Destructor:

```
PIRpRcResourceIdFact();
```

Default constructor

```
PIRpRcResourceIdFact(const PIRpRcResourceIdFact& orig);
```

Copy constructor, restricted copy

```
~PIRpRcResourceIdFact();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpRcResourceIdFact );
```

Instances of PIRpRcResourceIdFact are unique activity ID generators. You can ask them for the next allowable activity ID that can be assigned. This class initializes its activity ID range from the current activity pool for now. Max ID is presumed to be MAXINT.

- `init`

```
virtual int init(HString fldName, HString tblName);
```

Initialize. Set `myCurId`, `myMinId`, and `myMaxId` to some initial values, based upon the contents of the database. Max ID value is presumed to be MAXINT.

- `init`

```
virtual int init();
```

Initialize. Set `myCurId`, `myMinId`, and `myMaxId` to some initial values, based upon the contents of the current activity pool. Max ID value is presumed to be MAXINT.

- `nextResourceId`

```
static int nextResourceId();
```

Return the next available ID in the global PIRpRcResourceIdFact.

- `operator =`

```
PIRpRcResourceIdFact& operator =(const PIRpRcResourceIdFact&  
orig);
```

Assignment operator, restricted assignment

- `theResourceIdFact`

```
static int theResourceIdFact(PIRpRcResourceIdFact* idFactory);
```

Set the global PIRpRcResourceIdFact. This class does not assume ownership.

- `theResourceIdFact`

```
static PIRpRcResourceIdFact* theResourceIdFact();
```

Get the global PIRpRcResourceIdFact.

6.3.19.3.13 PIRpRcResource Class

Overview:

Instances of PIRpRcResource are base class for string, computer and disk resources

Export Control: Public

Inheritance Relationships:

Attributes:

`myActivity: HString`
Activity associated with this resource

Constructors and Destructor:

`PlRpRcResource();`
Default constructor

`PlRpRcResource(const PlRpRcResource& orig);`
Copy constructor

`virtual ~PlRpRcResource();`
Destructor

Operations:

- activity
`virtual const HString& activity() const;`
Get activity associated with this resource
- activity
`virtual int activity(const char* act);`
Set activity associated with this resource
- DECLARE_CLASS
`int DECLARE_CLASS(PlRpRcResource);`
Instances of PlRpRcResource are base class for string, computer and disk resources
- get
`virtual int get(istream& str);`
Get resource from stream
- operator =
`PlRpRcResource& operator =(const PlRpRcResource& orig);`
Assignment operator

- put

```
virtual int put(ostream& str) const;
```

Put resource to stream

- rsStateType

```
virtual HClassDesc& rsStateType();
```

Return the class description of my type of resource state. Default is to just return generic RRsState type. Derived classes should override to return a specialized type

- xdr

```
virtual int xdr(XDR* xdrs);
```

Get/put resource onto a stream

6.3.19.3.14 PIRpRcState Class

Overview:

class PIRpRcState (generic resource state)

Instances of PIRpRcState are generic activity states which simply contains the name of the generic state

Export Control: Public

Inheritance Relationships:

Attributes:

```
myName: HString
```

My state name

Constructors and Destructor:

```
PIRpRcState(const HEpochInterval& intvl, int actId, const char* name);
```

Constructor with interval, activity id and name specified

```
PIRpRcState(const PIRpRcState& orig);
```

Copy constructor

```
PIRpRcState(const HEpochInterval& intvl, int actId);
```

Constructor with interval and activity id

```
PlRpRcState();
```

Default constructor

```
virtual ~PlRpRcState();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpRcState );
```

class PlRpRcState (generic resource state)

Instances of PlRpRcState are generic activity states which simply contains the name of the generic state

- get

```
virtual int get(istream& istr);
```

Get state object from a stream

- name

```
virtual int name(const char* name);
```

Set my name

- name

```
virtual const HString& name() const;
```

Get my name

- operator =

```
PlRpRcState& operator =(const PlRpRcState& orig);
```

Assignment operator

- put

```
virtual int put(ostream& ostr) const;
```

Put state object onto a stream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get state object from a XDR stream

6.3.19.3.15 PIRpRcStorageState Class

Overview:

Instances of PIRpRcStorageState are states for storage utilization for a disk.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcMultiActState

Attributes:

myStorage: int

The storage consumed over my time interval

Constructors and Destructor:

```
PIRpRcStorageState(const PIRpRcStorageState& orig);
```

Copy constructor

```
PIRpRcStorageState(const HEpochInterval& );
```

Constructor with interval specified

```
PIRpRcStorageState();
```

Default constructor

```
virtual ~PIRpRcStorageState();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpRcStorageState );
```

Instances of PIRpRcStorageState are states for storage utilization for a disk.

- get

```
int get(istream& );
```

Get object from a stream

- operator =

```
PIRpRcStorageState& operator =(const PIRpRcStorageState& orig);
```

Assignment operator

- put

```
int put(ostream& ) const;
```

Put object onto a stream

- storage

```
virtual int storage() const;
```

Get the total storage consumed

- storage

```
virtual int storage(int newStorage);
```

Set the total storage consumed

- xdr

```
int xdr(XDR* );
```

Set/get object from a XDR stream

6.3.19.3.16 PIRpRcString Class

Overview:

Instances of PIRpRcString are logical groupings of computers

Export Control: Public

Inheritance Relationships:

Inherits from PIRpRcResource

Attributes:

myComputerIds: HVList

My list of Computer ids corresponding to the above list of Computers. I may have ids without having Computers if I have just been loaded from a stream (see resolveComputerIds() above).

myComputers: HObjList

List of associated Computers with this string resource

Constructors and Destructor:

```
PIRpRcString();
```

Default constructor

```
PIRpRcString(const PIRpRcString& orig);
```

Copy constructor

```
virtual ~PlRpRcString();
```

Destructor

Operations:

- addAssoc

```
virtual int addAssoc(HObject* rs);
```

Add an association to this string resource.

- addComputer

```
virtual int addComputer(RResource* computer);
```

Add a computer to this string.

- computers

```
virtual HObjIter* computers();
```

Return an iterator over my Computer list. The consumer must delete the iterator when done.

- copyOver

```
virtual int copyOver(const PlRpRcString& orig);
```

Copy over this string resource

- copyOver

```
virtual int copyOver(const RResource& orig);
```

Copy over this resource

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpRcString );
```

Instances of PlRpRcString are logical groupings of computers

- get

```
int get(istream& );
```

Get string resource from stream

- operator =

```
PlRpRcString& operator =(const PlRpRcString& orig);
```

Assignment operator

- put

```
int put(ostream& ) const;
```

Put string resource to stream

- removeComputer

```
virtual int removeComputer(RResource* computer);
```

Remove a computer from this string.

- resolveComputerIds

```
virtual void resolveComputerIds();
```

Try to resolve any Computer ids to Computers from the resource pool. Computers are passed around (ipc, to/from streams) using their reference resource id number. I may know a bunch of ids without yet having pointers to resources. This behavior will resolve any unknown ids by attempting to look them up in the RResourcePool.

- xdr

```
int xdr(XDR* );
```

Get/put string resource onto a stream

6.3.20 Resource_Planning_Re Class Category

6.3.20.1 Overview

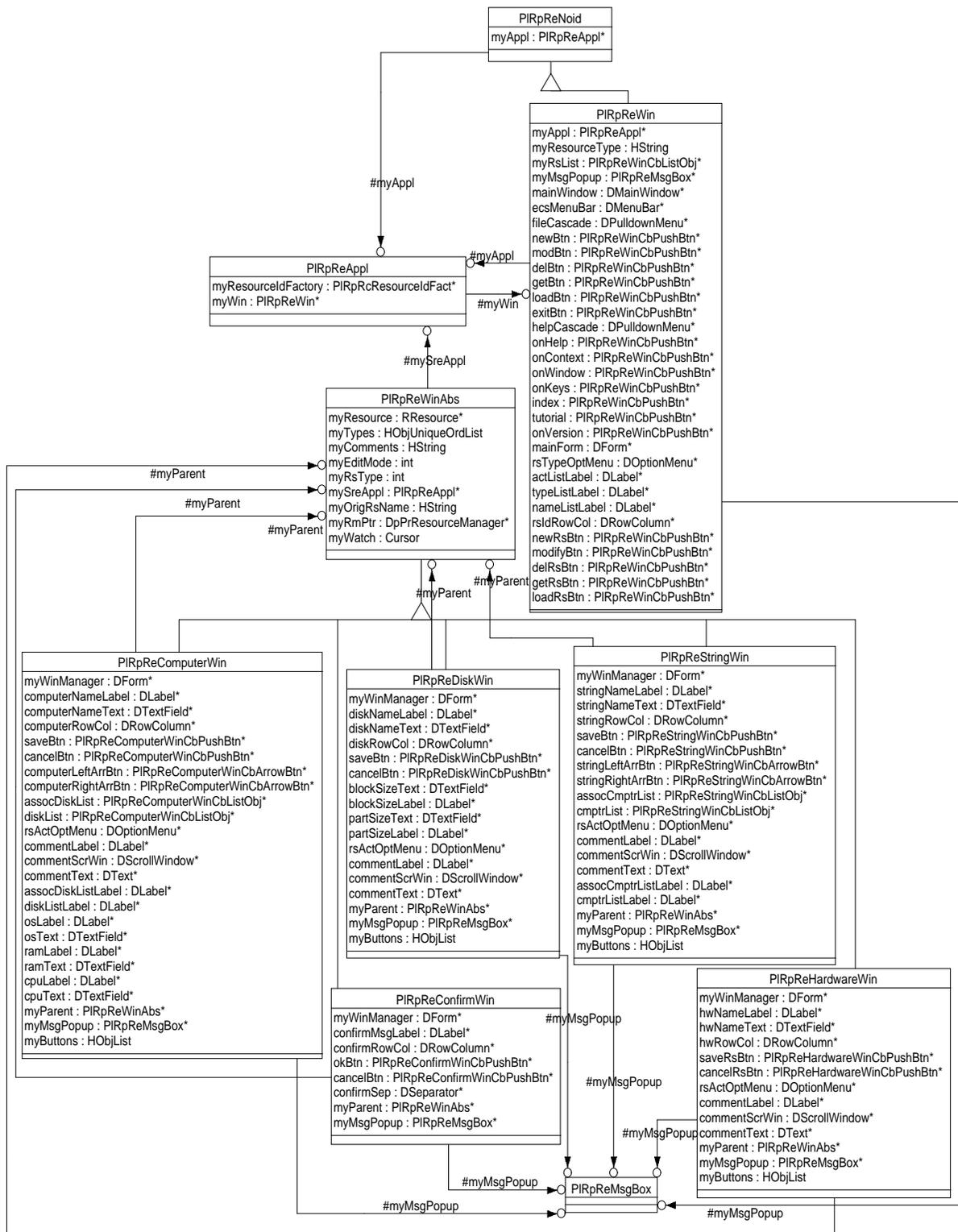


Figure 6.3.20.1-1 Resource_Planning_Re

6.3.20.2 Resource_Planning_Re Classes

6.3.20.3 PIRpReAppl Class

Overview:

Instances of PIRpReAppl are derived PIRpDiAppl's capable of using it's noid to request resources from the resource model.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpDiAppl

Attributes:

`myResourceIdFactory: PIRpRcResourceIdFact*`

The resource id factory

`myWin: PIRpReWin*`

Pointer to my window

Constructors and Destructor:

`PIRpReAppl(const PIRpReAppl& orig);`

Copy constructor

`PIRpReAppl();`

Default constructor

`virtual ~PIRpReAppl();`

Destructor

Operations:

- agent

`PIIpAgent* agent();`

Gets PIIpAgent

- connectionMade

`virtual int connectionMade();`

Do the remaining initialization when a connection is made

- createEnviron

```
virtual int createEnviron();
```

Set up the environment

- createIdFactories

```
virtual int createIdFactories();
```

Create the ID generator factories

- createMshNoid

```
virtual int createMshNoid();
```

Create a connection to the message handler.

- createResource

```
virtual int createResource(RResource* rs);
```

Ask the SRM to create a new resource.

- create

```
virtual int create(const char* appClass, int& argc, char** argv,  
XrmOptionDescList options, Cardinal numOptions);
```

Create Resource Editor Application

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpReAppl );
```

Instances of PIRpReAppl are derived PIRpDiAppl's capable of using it's noid to request resources from the resource model.

- delResource

```
virtual int delResource(RResource* rs);
```

Ask the SRM to delete a resource.

- initCommunications

```
virtual int initCommunications();
```

Initialize communications. Override PIRpCIAppl behavior to delay creation of SRM noid.

- init

```
virtual int init();
```

Initialize the Resource Editory Application

- makeWindow

```
virtual DWindow* makeWindow();
```

Make a production planning workbench Window

- modifyResource

```
virtual int modifyResource(RResource* rs);
```

Ask the SRM to modify a resource.

- newSrmNoid

```
virtual PIRpClSrmNoid* newSrmNoid();
```

Create the planning workbench SRM noid.

- operator =

```
PIRpReAppl& operator =(const PIRpReAppl& orig);
```

Assignment operator

- shutDown

```
virtual void shutDown();
```

Clean up pointers

6.3.20.3.1 PIRpReComputerWin Class

Overview:

Class PIRpReComputerWin (Show Computer (Host) Details Window). An instance of this class provides a view of the details associated with the selected computer (host) resource.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpReWinAbs

Attributes:

```
assocDiskListLabel: DLabel*
```

Associated disks list label widget

```
assocDiskList: PIRpReComputerWinCbListObj*
```

Scroll list widget containing list of disks associated with the computer

```
cancelBtn: PIRpReComputerWinCbPushBtn*
```

Cancel push button widget

```
commentLabel: DLabel*
```

Comment label widget

commentScrWin: DScrollWindow*
Scroll window widget for comments text field

commentText: DText*
Text field widget for comments

computerLeftArrBtn: PlRpReComputerWinCbArrowBtn*
Left arrow button widget

computerNameLabel: DLabel*
Computer name label widget

computerNameText: DTextField*
Computer name text field widget

computerRightArrBtn: PlRpReComputerWinCbArrowBtn*
Right arrow button widget

computerRowCol: DRowColumn*
Row column containing push button widgets

cpuLabel: DLabel*
Number of CPUs label widget

cpuText: DTextField*
Number of CPUs text field widget

diskListLabel: DLabel*
Disk list label widget

diskList: PlRpReComputerWinCbListObj*
Scroll list widget containing list of all disks

myButtons: HObjList
List of push button widgets containing names of activities in option menu

myMsgPopup: PlRpReMsgBox*
Pointer to message box popup

myParent: PlRpReWinAbs*
Pointer to parent window

myWinManager: DForm*
Main form widget

osLabel: DLabel*

Operating system label widget

osText: DTextField*

Operating system text field widget

ramLabel: DLabel*

Total RAM label widget

ramText: DTextField*

Total RAM text field widget

rsActOptMenu: DOptionMenu*

Option menu widget containing list of activities

saveBtn: PlRpReComputerWinCbPushBtn*

Save push button widget

Constructors and Destructor:

PlRpReComputerWin(PlRpReWinAbs* parent);

Constructor with visibility to parent window.

virtual ~PlRpReComputerWin();

Default Destructor.

Operations:

- addLocalDisks

virtual void addLocalDisks(PlRpRcComputer* cRs);

Routine to add selected local disks to a computer

- assocListCb

**virtual void assocListCb(PlRpReComputerWinCbListObj* list,
XtPointer callData);**

Callback - Routine to move selected items from disk list to associated disk list

- buildArrowBtns

virtual int buildArrowBtns();

Build arrow button widgets

- buildButtons

```
virtual int buildButtons();
```

Build push button widgets

- buildLabels

```
virtual int buildLabels();
```

Build label widgets

- buildOptMenus

```
virtual int buildOptMenus();
```

Build option menu widgets

- buildRowCols

```
virtual int buildRowCols();
```

Build row column widgets

- buildScrLists

```
virtual int buildScrLists();
```

Build scrolled list widgets

- buildScrText

```
virtual int buildScrText();
```

Build scrolled text field widgets

- buildTextFields

```
virtual int buildTextFields();
```

Build text field widgets

- buildWindow

```
virtual int buildWindow();
```

Build main window

- cancelBtnCb

```
virtual void cancelBtnCb(PlRpReComputerWinCbPushBtn* btn,  
XtPointer callData);
```

Callback - Popdown and destroy window, when no longer needed.

- cleanup

```
virtual void cleanup();
```

Override base class member to provide window specific cleanup needs.

- createDisplay

```
virtual int createDisplay();
```

Create display

- createMsgBox

```
virtual int createMsgBox();
```

Create message box popup

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpReComputerWin );
```

Class PlRpReComputerWin (Show Computer (Host) Details Window). An instance of this class provides a view of the details associated with the selected computer (host) resource.

- diskListCb

```
virtual void diskListCb(PlRpReComputerWinCbListObj* list,  
XtPointer callData);
```

Callback - Routine to move selected items from associated disk list to disk list

- fillActList

```
virtual int fillActList();
```

Fill activity option menu with each unique resource activity found in the local resource pool after configuration.

- fillAssocDiskList

```
virtual int fillAssocDiskList(PlRpRcComputer* );
```

Fill associated disk list with list of disks associated with the computer

- fillDiskList

```
virtual int fillDiskList();
```

Fill disk list with list of all disks

- init

```
virtual int init();
```

Override initialization from base class to provide specific window setup.

- leftArrBtnCb

```
virtual void leftArrBtnCb(PlRpReComputerWinCbArrowBtn* btn,  
XtPointer callData);
```

Callback - Move selected item from associated disk list to disk list

- moveSelections

```
virtual void moveSelections(DList* fromList, DList* toList);
```

Move selected items from one list to other

- removeLocalDisks

```
virtual void removeLocalDisks(PIRpRcComputer* cRs);
```

Routine to remove all local disks from a computer

- rightArrBtnCb

```
virtual void rightArrBtnCb(PIRpReComputerWinCbArrowBtn* btn,
XtPointer callData);
```

Callback - Move selected item from disk list to associated disk list

- saveBtnCb

```
virtual void saveBtnCb(PIRpReComputerWinCbPushBtn* btn,
XtPointer callData);
```

Callback - Save the comments to database

- updateView

```
virtual int updateView(const char* token);
```

Update my view when the model changes in the way described by token.

6.3.20.3.2 PIRpReConfirmWin Class

Overview:

Class PIRpReConfirmWin (Resource Editor Delete Confirmation Window) An instance of this class provides a confirmation window for deleting resources

Export Control: Public

Inheritance Relationships:

Inherits from PIRpReWinAbs

Attributes:

```
cancelBtn: PIRpReConfirmWinCbPushBtn*
```

Cancel push button widget

```
confirmMsgLabel: DLabel*
```

Confirmation message label widget

confirmRowCol: DRowColumn*

Row column containing push button widgets

confirmSep: DSeparator*

Separator widget

myMsgPopup: PlRpReMsgBox*

Pointer to message box popup

myParent: PlRpReWinAbs*

Pointer to parent window

myWinManager: DForm*

Main form widget

okBtn: PlRpReConfirmWinCbPushBtn*

OK push button widget

Constructors and Destructor:

```
PlRpReConfirmWin(PlRpReWinAbs* parent);
```

Constructor.

```
virtual ~PlRpReConfirmWin();
```

Default Destructor.

Operations:

- buildButtons

```
virtual int buildButtons();
```

Build push button widgets

- buildLabels

```
virtual int buildLabels();
```

Build label widgets

- buildRowCols

```
virtual int buildRowCols();
```

Build row column widgets

- buildSeps

```
virtual int buildSeps();
```

Build separator widgets

- buildWindow

```
virtual int buildWindow();
```

Build main window

- cancelBtnCb

```
virtual void cancelBtnCb(PIRpReConfirmWinCbPushBtn* btn,  
XtPointer callData);
```

Callback - Popdown and destroy window, when no longer needed.

- cleanup

```
virtual void cleanup();
```

Override base class member to provide window specific cleanup needs.

- createDisplay

```
virtual int createDisplay();
```

Create display

- createMsgBox

```
virtual int createMsgBox();
```

Create message box popup

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpReConfirmWin );
```

Class PIRpReConfirmWin (Resource Editor Delete Confirmation Window) An instance of this class provides a confirmation window for deleting resources

- init

```
virtual int init();
```

Override initialization from base class to provide specific window setup.

- okBtnCb

```
virtual void okBtnCb(PIRpReConfirmWinCbPushBtn* btn, XtPointer  
callData);
```

Callback - Save the comments to database

6.3.20.3.3 PIRpReDiskWin Class

Overview:

Class PIRpReDiskWin (Show Disk Partition Details Window) An instance of this class provides a view of the details associated with the selected disk partition resource.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpReWinAbs

Attributes:

blockSizeLabel: DLabel*

Block size label widget

blockSizeText: DTextField*

Block size text field widget

cancelBtn: PIRpReDiskWinCbPushBtn*

Cancel push button widget

commentLabel: DLabel*

Comment label widget

commentScrWin: DScrollWindow*

Scroll window widget for comments text field

commentText: DText*

Text field widget for comments

diskNameLabel: DLabel*

Disk name label widget

diskNameText: DTextField*

Disk name text field widget

diskRowCol: DRowColumn*

Row column containing push button widgets

myButtons: HObjList

List of push button widgets containing names of activities in option menu

myMsgPopup: PIRpReMsgBox*

Pointer to message box popup

myParent: PIRpReWinAbs*

Pointer to parent window

myWinManager: DForm*

Main form widget

partSizeLabel: DLabel*

Partition size label widget

partSizeText: DTextField*

Partition size text field widget

rsActOptMenu: DOptionMenu*

Option menu widget containing list of activities

saveBtn: PIRpReDiskWinCbPushBtn*

Save push button widget

Constructors and Destructor:

PIRpReDiskWin(PIRpReWinAbs* parent);

Constructor with visibility to parent window.

virtual ~PIRpReDiskWin();

Default Destructor.

Operations:

- buildButtons

virtual int buildButtons();

Build push button widgets

- buildLabels

virtual int buildLabels();

Build label widgets

- buildOptMenus

virtual int buildOptMenus();

Build option menu widgets

- buildRowCols

```
virtual int buildRowCols();
```

Build row column widgets

- buildScrText

```
virtual int buildScrText();
```

Build scrolled text field widgets

- buildTextFields

```
virtual int buildTextFields();
```

Build text field widgets

- buildWindow

```
virtual int buildWindow();
```

Build main window

- cancelBtnCb

```
virtual void cancelBtnCb(PIRpReDiskWinCbPushBtn* btn, XtPointer  
callData);
```

Callback - Popdown and destroy window, when no longer needed.

- cleanup

```
virtual void cleanup();
```

Override base class member to provide window specific cleanup needs.

- createDisplay

```
virtual int createDisplay();
```

Create display

- createMsgBox

```
virtual int createMsgBox();
```

Create message box popup

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpReDiskWin );
```

Class PIRpReDiskWin (Show Disk Partition Details Window) An instance of this class provides a view of the details associated with the selected disk partition resource.

- fillActList

```
virtual int fillActList();
```

Fill activity option menu with each unique resource activity found in the local resource pool after configuration.

- `init`

```
virtual int init();
```

Override initialization from base class to provide specific window setup.

- `saveBtnCb`

```
virtual void saveBtnCb(PIRpReDiskWinCbPushButton* btn, XtPointer  
callData);
```

Callback - Save the comments to database

- `updateView`

```
virtual int updateView(const char* token);
```

Update my view when the model changes in the way described by token.

6.3.20.3.4 PIRpReHardwareWin Class

Overview:

Class PIRpReHardwareWin (Show Hardware Details Window) An instance of this class provides a view of the details associated with the selected hardware resource.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpReWinAbs

Attributes:

```
cancelRsBtn: PIRpReHardwareWinCbPushButton*
```

Cancel push button widget

```
commentLabel: DLabel*
```

Comment label widget

```
commentScrWin: DScrollWindow*
```

Scroll window widget for comments text field

```
commentText: DText*
```

Text field widget for comments

```
hwNameLabel: DLabel*
```

Hardware name label widget

hwNameText: DTextField*

Hardware name text field widget

hwRowCol: DRowColumn*

Row column containing push button widgets

myButtons: HObjList

List of push button widgets containing names of activities in option menu

myMsgPopup: PlRpReMsgBox*

Pointer to message box popup

myParent: PlRpReWinAbs*

Pointer to parent window

myWinManager: DForm*

Main form widget

rsActOptMenu: DOptionMenu*

Option menu widget containing list of activities

saveRsBtn: PlRpReHardwareWinCbPushBtn*

Save push button widget

Constructors and Destructor:

```
PlRpReHardwareWin(PlRpReWinAbs* parent);
```

Constructor with visibility to parent window.

```
virtual ~PlRpReHardwareWin();
```

Default Destructor.

Operations:

- buildButtons

```
virtual int buildButtons();
```

Build push button widgets

- buildLabels

```
virtual int buildLabels();
```

Build label widgets

- buildOptMenus

```
virtual int buildOptMenus();
```

Build option menu widgets

- buildRowCols

```
virtual int buildRowCols();
```

Build row column widgets

- buildScrText

```
virtual int buildScrText();
```

Build scrolled text field widgets

- buildTextFields

```
virtual int buildTextFields();
```

Build text field widgets

- buildWindow

```
virtual int buildWindow();
```

Build main window

- cancelBtnCb

```
virtual void cancelBtnCb(PlRpReHardwareWinCbPushBtn* btn,  
XtPointer callData);
```

Callback - Popdown and destroy window, when no longer needed.

- cleanup

```
virtual void cleanup();
```

Override base class member to provide window specific cleanup needs.

- createDisplay

```
virtual int createDisplay();
```

Create display

- createMsgBox

```
virtual int createMsgBox();
```

Create message box popup

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpReHardwareWin );
```

Class PIRpReHardwareWin (Show Hardware Details Window) An instance of this class provides a view of the details associated with the selected hardware resource.

- fillActList

```
virtual int fillActList();
```

Fill activity option menu with each unique resource activity found in the local resource pool after configuration.

- init

```
virtual int init();
```

Override initialization from base class to provide specific window setup.

- saveBtnCb

```
virtual void saveBtnCb(PlRpReHardwareWinCbPushBtn* btn,  
XtPointer callData);
```

Callback - Save the comments to database

- updateView

```
virtual int updateView(const char* token);
```

Update my view when the model changes in the way described by token.

6.3.20.3.5 PIRpReMsgBox Class

Overview:

Class PIRpReMsgBox (Resource Editor Message Popup) An instance of this provides a message popup box for displaying information to the user

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PlRpReMsgBox();
```

Constructor.

```
virtual ~PlRpReMsgBox();
```

Default Destructor.

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpReMsgBox );
```

Class PlRpReMsgBox (Resource Editor Message Popup) An instance of this provides a message popup box for displaying information to the user

- init

```
virtual int init(DWidget* parent);
```

Initialize message box.

- show

```
virtual void show(const char* msg);
```

Popup message box with given message string.

6.3.20.3.6 PlRpReNoid Class

Overview:

Class PlRpReNoid (derived PlRpClSrmNoid) Instances of PlRpReNoids handle resource planning related client noid messages.

Export Control: Public

Inheritance Relationships:

Inherits from PlRpClSrmNoid

Attributes:

```
myAppl: PlRpReAppl*
```

Pointer to the Resource Editory application

Constructors and Destructor:

```
PlRpReNoid(PlRpReAppl* appl);
```

Constructor with application class

```
PlRpReNoid();
```

Default Constructor

```
PlRpReNoid(const PlRpReNoid& orig);
```

Copy constructor

```
virtual ~PIRpReNoid();
```

Destructor

Operations:

- appl

```
virtual void appl(PIRpReAppl* appl);
```

Give me visibility to my application.

- connEstablished

```
virtual int connEstablished();
```

When connection is established, initialize.

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpReNoid );
```

Class PIRpReNoid (derived PIRpClSrmNoid) Instances of PIRpReNoids handle resource planning related client noid messages.

- handleRsChg

```
virtual int handleRsChg(HMsgConn* con, HAddress* adr,  
PIRpClMsgRsChg* msg);
```

Method to handle derived modified resource message.

- handleRsDel

```
virtual int handleRsDel(HMsgConn* con, HAddress* adr,  
PIRpClMsgRsDel* msg);
```

Method to handle derived delete resource message.

- handleRsNew

```
virtual int handleRsNew(HMsgConn* con, HAddress* adr,  
PIRpClMsgRsNew* msg);
```

Method to handle derived new resource message.

- init

```
virtual int init();
```

Initialize the application.

- operator =

```
PlRpReNoid& operator =(const PlRpReNoid& orig);
```

Assignment operator

6.3.20.3.7 PIRpReStringWin Class

Overview:

Class PIRpReStringWin (Show String Details Window) An instance of this class provides a view of the details associated with the selected string resource.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpReWinAbs

Attributes:

assocCmptrListLabel: DLabel*

Associated computers list label widget

assocCmptrList: PlRpReStringWinCbListObj*

Scroll list widget containing list of computers associated with the string

cancelBtn: PlRpReStringWinCbPushBtn*

Cancel push button widget

cmptrListLabel: DLabel*

Computers list label widget

cmptrList: PlRpReStringWinCbListObj*

Scroll list widget containing list of all computers

commentLabel: DLabel*

Comment label widget

commentScrWin: DScrollWindow*

Scroll window widget for comments text field

commentText: DText*

Text field widget for comments

myButtons: HObjList

List of push button widgets containing names of activities in option menu

myMsgPopup: PlRpReMsgBox*

Pointer to message box popup

myParent: PlRpReWinAbs*

Pointer to parent window

myWinManager: DForm*

Main form widget

rsActOptMenu: DOptionMenu*

Option menu widget containing list of activities

saveBtn: PlRpReStringWinCbPushBtn*

Save push button widget

stringLeftArrBtn: PlRpReStringWinCbArrowBtn*

Left arrow button widget

stringNameLabel: DLabel*

String name label widget

stringNameText: DTextField*

String name text field widget

stringRightArrBtn: PlRpReStringWinCbArrowBtn*

Right arrow button widget

stringRowCol: DRowColumn*

Row column containing push button widgets

Constructors and Destructor:

```
PlRpReStringWin(PlRpReWinAbs* parent);
```

Constructor with visibility to parent window.

```
virtual ~PlRpReStringWin();
```

Default Destructor.

Operations:

- addComputers

```
virtual void addComputers(PlRpRcString* sRs);
```

Routine to add selected computers to a string

- assocListCb

```
virtual void assocListCb(PlRpReStringWinCbListObj* list,  
XtPointer callData);
```

Routine to move selected items from disk list to associated disk list

- buildArrowBtns

```
virtual int buildArrowBtns();
```

Build arrow button widgets

- buildButtons

```
virtual int buildButtons();
```

Build push button widgets

- buildLabels

```
virtual int buildLabels();
```

Build label widgets

- buildOptMenus

```
virtual int buildOptMenus();
```

Build option menu widgets

- buildRowCols

```
virtual int buildRowCols();
```

Build row column widgets

- buildScrLists

```
virtual int buildScrLists();
```

Build scrolled list widgets

- buildScrText

```
virtual int buildScrText();
```

Build scrolled text field widgets

- buildTextFields

```
virtual int buildTextFields();
```

Build text field widgets

- buildWindow

```
virtual int buildWindow();
```

Build main window

- cancelBtnCb

```
virtual void cancelBtnCb(PlRpReStringWinCbPushBtn* btn,  
XtPointer callData);
```

Callback - Popdown and destroy window, when no longer needed.

- cleanup

```
virtual void cleanup();
```

Override base class member to provide window specific cleanup needs.

- cmptrListCb

```
virtual void cmptrListCb(PlRpReStringWinCbListObj* list,  
XtPointer callData);
```

Routine to move selected items from associated disk list to disk list

- createDisplay

```
virtual int createDisplay();
```

Create display

- createMsgBox

```
virtual int createMsgBox();
```

Create message box popup

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpReStringWin );
```

Class PlRpReStringWin (Show String Details Window) An instance of this class provides a view of the details associated with the selected string resource.

- fillActList

```
virtual int fillActList();
```

Fill activity option menu with each unique resource activity found in the local resource pool after configuration.

- fillAssocCmptrList

```
virtual int fillAssocCmptrList(PlRpRcString* );
```

Fill associated computer list with list of computers associated with the string

- fillCmptrList

```
virtual int fillCmptrList();
```

Fill computer list with list of all computers

- `init`

```
virtual int init();
```

Override initialization from base class to provide specific window setup.

- `leftArrBtnCb`

```
virtual void leftArrBtnCb(PlRpReStringWinCbArrowBtn* btn,  
XtPointer callData);
```

Move selected item from associated disk list to disk list

- `moveSelections`

```
virtual void moveSelections(DList* fromList, DList* toList);
```

Move selected items from one list to other

- `removeComputers`

```
virtual void removeComputers(PlRpRcString* sRs);
```

Routine to remove all computers from a string

- `rightArrBtnCb`

```
virtual void rightArrBtnCb(PlRpReStringWinCbArrowBtn* btn,  
XtPointer callData);
```

Move selected item from disk list to associated disk list

- `saveBtnCb`

```
virtual void saveBtnCb(PlRpReStringWinCbPushBtn* btn, XtPointer  
callData);
```

Callback - Save the string resource to database

- `updateView`

```
virtual int updateView(const char* token);
```

Update my view when the model changes in the way described by token.

6.3.20.3.8 PIRpReWinAbs Class

Overview:

Class PIRpReWinAbs (Abstract Resource Editor Window) PIRpReWinAbs establishes protocol for System Resource Editor Windows.

Export Control: Public

Inheritance Relationships:

Attributes:

myComments: HString

Resource comments

myEditMode: int

Edit mode

myOrigRsName: HString

The original name of the resource (when modifying)

myResource: RResource*

Visibility to my associated resource.

myRmPtr: DpPrResourceManager*

Pointer to Resource Manager (for locking tables)

myRsType: int

Resource type

mySreAppl: PlRpReAppl*

Pointer to the application

myTypes: HObjUniqueOrdList

List of activity types

myWatch: Cursor

Busy cursor

Constructors and Destructor:

PlRpReWinAbs(const PlRpReWinAbs& orig);

Copy constructor

PlRpReWinAbs();

Default Constructor.

virtual ~PlRpReWinAbs();

Destructor.

Operations:

- checkForReservation

```
virtual int checkForReservation(RResource* rs);
```

Check to see if resource is reserved

- cleanup

```
virtual void cleanup();
```

Delete window and tell sre that I'm going away.

- clearDep

```
virtual int clearDep(HObject* obj);
```

Clear view dependency on the desired object. Default behavior is to clear dependency on my associated resource.

- DECLARE_ABS_CLASS

```
int DECLARE_ABS_CLASS(PIRpReWinAbs );
```

Class PIRpReWinAbs (Abstract Resource Editor Window) PIRpReWinAbs establishes protocol for System Resource Editor Windows.

- edit_mode

```
virtual int edit_mode(enum PIRpReWinAbs::EditMode eMode);
```

Set my edit mode.

- edit_mode

```
virtual int edit_mode() const;
```

Get my edit mode.

- getActivityId

```
virtual int getActivityId(HString actName);
```

Get the activity id for the selected activity

- getActList

```
virtual int getActList();
```

Fill activity option menu with each unique resource activity found in the local resource pool after configuration (and the defaults).

- getComments

```
virtual int getComments(int rsId);
```

Retrieve comments from database

- `initIdFact`
`virtual int initIdFact();`
 Initializes factory id
- `init`
`virtual int init();`
 Initialize window. Derived classes will override this to provide application specific window setup.
- `lockResourceTbls`
`virtual int lockResourceTbls(RResource* rrs);`
 Lock resource tables before updating (for Resource Management)
- `makeDep`
`virtual int makeDep(HObject* obj);`
 Make this view dependent on the desired object. Default behavior is to make window dependent on my associated resource.
- `notifySRM`
`virtual int notifySRM(RResource* rs, int editmode);`
 Notify SRM of new/changed resource
- `operator =`
`PlRpReWinAbs& operator =(const PlRpReWinAbs& orig);`
 Assignment operator
- `origRsName`
`virtual int origRsName(const char* rsName);`
 Set original resource name (when modifying a resource)
- `origRsName`
`virtual const HString& origRsName() const;`
 Get original resource name (when modifying a resource)
- `resource`
`virtual RResource* resource() const;`
 Get my associated resource.
- `resource`

```
virtual int resource(RResource* rs);
```

Set my associated resource.

- selectedRsName

```
virtual HString* selectedRsName(DList* selList);
```

Get the source name currently selected

- setWatch

```
virtual void setWatch(int flag);
```

Set my cursor to watch (if TRUE) or back to normal (if FALSE). overridden from base class to avoid core dump when there is no window.

- sreAppl

```
PIRpReAppl* sreAppl(PIRpReAppl* theAppl);
```

Set the sre application instance

- sreAppl

```
PIRpReAppl* sreAppl();
```

Get the sre application instance

- unlockResourceTbls

```
virtual int unlockResourceTbls(int theRsId);
```

Unlock resource tables after updating (for Resource Management)

6.3.20.3.9 PIRpReWin Class

Overview:

Class PIRpReWin (A derived DWindow) This class is Motif Window for controlling the setup and display of the production planning workbench.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpReNoid

Attributes:

```
actListLabel: DLabel*
```

Activity list label widget

```
delBtn: PIRpReWinCbPushBtn*
```

Delete button widget

delRsBtn: PlRpReWinCbPushBtn*

Delete push button widget

ecsMenuBar: DMenuBar*

Menu bar widget

exitBtn: PlRpReWinCbPushBtn*

Exit button widget

fileCascade: DPulldownMenu*

File pulldown menu widget

getBtn: PlRpReWinCbPushBtn*

Fetch resources button widget

getRsBtn: PlRpReWinCbPushBtn*

Fetch resources push button widget

helpCascade: DPulldownMenu*

Help pulldown menu widget

index: PlRpReWinCbPushBtn*

Index button widget

loadBtn: PlRpReWinCbPushBtn*

Load resources button widget

loadRsBtn: PlRpReWinCbPushBtn*

Load resources push button widget

mainForm: DForm*

Main form widget

mainWindow: DMainWindow*

Main window widget

modBtn: PlRpReWinCbPushBtn*

Modify button widget

modifyBtn: PlRpReWinCbPushBtn*

Modify push button widget

myAppl: PlRpReAppl*

Pointer to Resource Editor application

myMsgPopup: PlRpReMsgBox*

Pointer to message box popup

myResourceType: HString

Selected type of resource

myRsList: PlRpReWinCbListObj*

Scroll list widget containing list of resources

nameListLabel: DLabel*

Resource name label widget

newBtn: PlRpReWinCbPushBtn*

New button widget

newRsBtn: PlRpReWinCbPushBtn*

New push button widget

onContext: PlRpReWinCbPushBtn*

On Context button widget

onHelp: PlRpReWinCbPushBtn*

On help button widget

onKeys: PlRpReWinCbPushBtn*

On Keys button widget

onVersion: PlRpReWinCbPushBtn*

On version button widget

onWindow: PlRpReWinCbPushBtn*

On Window button widget

rsIdRowCol: DRowColumn*

Row column containing push button widgets

rsTypeOptionsMenu: DOptionMenu*

Option menu widget containing list of resource types

tutorial: PlRpReWinCbPushBtn*

Tutorial button widget

typeListLabel: DLabel*

Resource type list label widget

Constructors and Destructor:

```
PlRpReWin(const PlRpReWin& orig);
```

Copy constructor

```
PlRpReWin(PlRpReAppl* appl);
```

Default Constructor

```
virtual ~PlRpReWin();
```

Default Destructor

Operations:

- buildButtons

```
virtual int buildButtons();
```

Build push button widgets

- buildLabels

```
virtual int buildLabels();
```

Build label widgets

- buildMenuBar

```
virtual int buildMenuBar();
```

Create menu bar widgets

- buildOptMenus

```
virtual int buildOptMenus();
```

Build option menu widgets

- buildRowCols

```
virtual int buildRowCols();
```

Build row column widgets

- buildRsList

```
virtual int buildRsList();
```

Build scroll list widgets

- createComputerWin

```
virtual PlRpReComputerWin* createComputerWin();
```

Instantiate a window for creating/modifying computer resources

- createConfirmWin

```
virtual PlRpReConfirmWin* createConfirmWin();
```

Instantiate a window for displaying delete confirmation

- createDiskWin

```
virtual PlRpReDiskWin* createDiskWin();
```

Instantiate a window for creating/modifying disk resources

- createDisplay

```
virtual int createDisplay();
```

Overriden from base class to provide window display creation.

- createHardwareWin

```
virtual PlRpReHardwareWin* createHardwareWin();
```

Instantiate a window for creating/modifying hardware resources

- createMsgBox

```
virtual int createMsgBox();
```

Create message box popup.

- createStringWin

```
virtual PlRpReStringWin* createStringWin();
```

Instantiate a window for creating/modifying string resources

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpReWin );
```

Class PlRpReWin (A derived DWindow) This class is Motif Window for controlling the setup and display of the production planning workbench.

- delBtnCb

```
virtual void delBtnCb(PlRpReWinCbPushBtn* , XtPointer cbs);
```

Callback - Delete selected resource

- exitBtnCb

```
virtual void exitBtnCb(PlRpReWinCbPushBtn* , XtPointer cbs);
```

Callback - Exit application

- fillTypeList

```
virtual int fillTypeList();
```

Fill type option menu with each unique resource type found in the local resource pool after configuration.

- getRsBtnCb

```
virtual void getRsBtnCb(PlRpReWinCbPushButton* , XtPointer cbs);
```

Callback - Fetch resource from Baseline Manager (MSS)

- initIcon

```
virtual void initIcon();
```

Overriden from base class to set my personal icon.

- init

```
virtual int init();
```

Initialize window

- loadRsBtnCb

```
virtual void loadRsBtnCb(PlRpReWinCbPushButton* , XtPointer cbs);
```

Callback - load resources from configuration file retrieved from Baseline Manager (MSS) (after Fetch baseline button pushed)

- modifyBtnCb

```
virtual void modifyBtnCb(PlRpReWinCbPushButton* , XtPointer cbs);
```

Callback - Modify selected resource

- newBtnCb

```
virtual void newBtnCb(PlRpReWinCbPushButton* , XtPointer cbs);
```

Callback - Create a new resource

- operator =

```
PlRpReWin& operator =(const PlRpReWin& orig);
```

Assignment operator

- padStr

```
virtual HString& padStr(HString& str, int len);
```

Routine to right pad an HString with spaces

- parseCommandLine

```
virtual int parseCommandLine();
```

Parse the command line.

- rsSelected

```
virtual int rsSelected();
```

Determine if user has selected a resource list item. Returns TRUE if selected; FALSE if not.

- selectedRs

```
virtual RResource* selectedRs();
```

Get the resource associated with the selected resource from the local pool.

- selectListCb

```
virtual void selectListCb(PlRpReWinCbListObj* list, XtPointer  
cbs);
```

Callback - Toggle list selection

- setTypeCb

```
virtual void setTypeCb(PlRpReWinCbPushBtn* , XtPointer cbs);
```

Callback - Set resource type

- setup

```
virtual int setup();
```

Setup display.

- updateRsList

```
virtual int updateRsList();
```

Update the resource display list from the local resource pool.

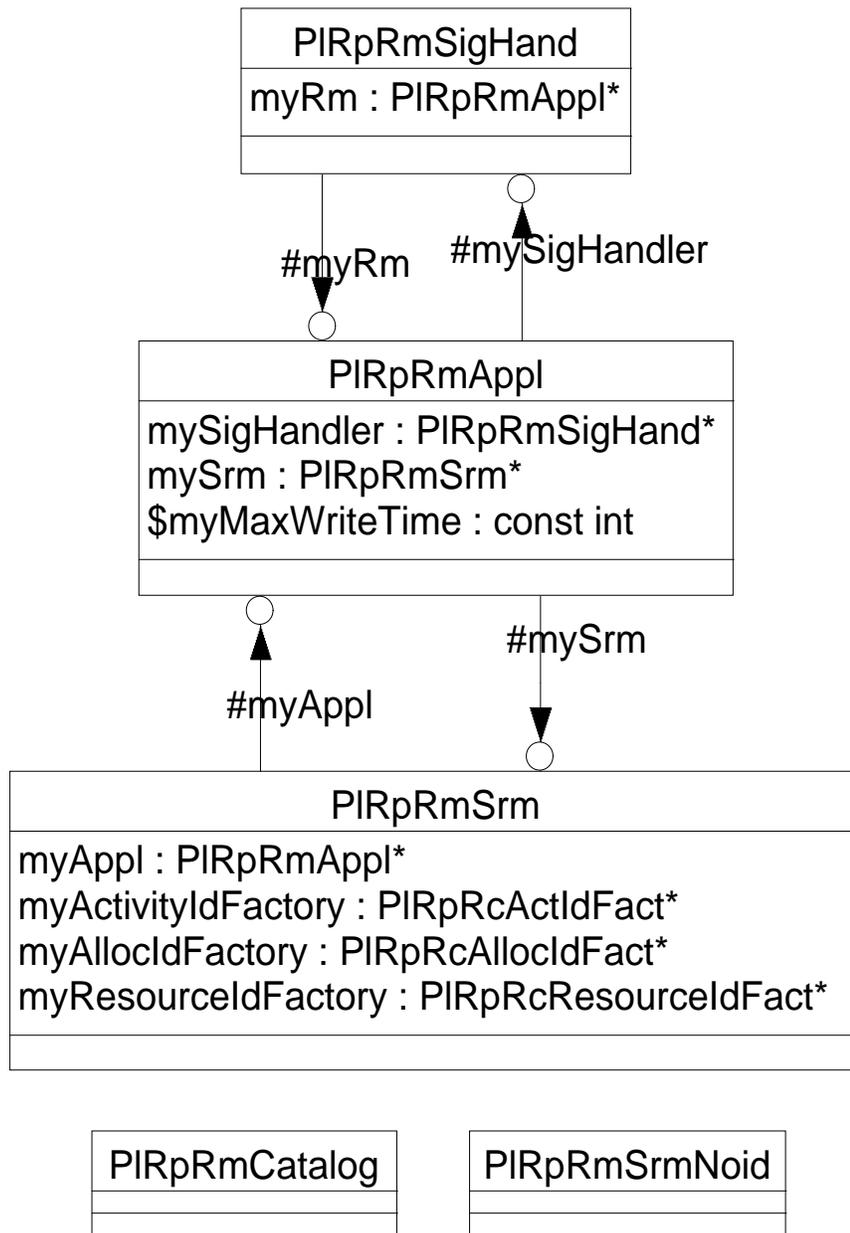
- update

```
virtual int update();
```

Update window

6.3.21 Resource_Planning_Rm Class Category

6.3.21.1 Overview



6.3.21.2 Resource_Planning_Rm Classes

6.3.21.3 PIRpRmAppl Class

Overview:

Instances of PIRpRmAppl are resource model applications. PIRpRmAppl is responsible for creating and/or connecting to the name server, message agent, resource and activity pools, and scheduling noids. PIRpRmAppl is also responsible for setting up the notifier for running the resource model application.

Export Control: Public

Inheritance Relationships:

Inherits from PIMiAppl

Attributes:

myMaxWriteTime: const int

The ipc write timeout time

mySigHandler: PlRpRmSigHand*

The signal handler

mySrm: PlRpRmSrm*

The resource model workhorse to handle client requests

Constructors and Destructor:

PlRpRmAppl();

Default constructor

PlRpRmAppl(const PlRpRmAppl& orig);

Copy constructor

virtual ~PlRpRmAppl();

Destructor

Operations:

- createActPool

virtual int createActPool();

Create the activity pool

- createAgent

virtual int createAgent();

Create an instance of a HMsgAgent. Overrides PIMiAppl::createAgent to set the connection's maximum write time

- createMshNoid

virtual int createMshNoid();

Create a connection to the message handler.

- createPlanPool

virtual int createPlanPool();

Create the plan pool

- createPools

virtual int createPools();

Create the pools that the SRM needs

- createRsPool

virtual int createRsPool();

Create the resource pool

- createSchedulableResources

virtual int createSchedulableResources() const;

Create new schedulable resources (PIRpRcResource) and add them to the schedulable resource pool.

- createSigHandler

virtual int createSigHandler();

Create a signal handler who will call the SRM when certain signals are received

- createSrmNoid

virtual int createSrmNoid();

Create my message noid.

- createSrm

virtual int createSrm();

Create the resource model main class. This is an instance of an PIRpRmSrm object which has scheduling behavior in response to external input.

- createSrsPool

virtual int createSrsPool();

Create the SResource pool

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpRmAppl );
```

Instances of PlRpRmAppl are resource model applications. PlRpRmAppl is responsible for creating and/or connecting to the name server, message agent, resource and activity pools, and scheduling noids. PlRpRmAppl is also responsible for setting up the notifier for running the resource model application.

- init

```
virtual int init();
```

Initialize this application. Overrides PlMiAppl::init to initialize those things I need.

- loadPools

```
virtual int loadPools() const;
```

Fill the various pools with their static and dynamic states. Currently, the state initialization is done from a database.

- operator =

```
PlRpRmAppl& operator =(const PlRpRmAppl& orig);
```

Assignment operator

- printFlags

```
virtual void printFlags();
```

Dump out a description of the run-time flags required to run this application. Overrides PlMiAppl::printFlags to print out more flags needed by SRM

- printVerbage

```
virtual void printVerbage();
```

Dump out a usage message as to why this appl exists. Overrides PlMiAppl::printVerbage to print out what I am/do

- saveAllocations

```
virtual int saveAllocations(const char* plan, const  
HEpochInterval& interval, const HVList* rsIds);
```

Save the activity allocations given the resources, plan and time interval. If no resources are specified, all resources are saved. If no plan is specified, all plans are saved, if no interval is set, the entire time span is saved.

- saveAll

```
virtual int saveAll();
```

Note: Portions of the plan may be saved/opened by specifying the resources to save on a specific plan over a specific interval of time. If the rs id collection is empty all resource states and allocations associated with all resources are saved/opened. If the plan is not specified, all plans are saved/opened. If the interval is not set than the entire time span is saved/opened

Save all my state

- savePlan

```
virtual int savePlan(PlPlPlan* plan);
```

Save a plan, this should be removed when the plan manager and messages are in place to save an actual plan objects themselves Note: If the plan is NULL, all plans in the resource model will be saved

- savePlan

```
virtual int savePlan(const char* plan, const HEpochInterval& interval, const HVList* rsIds);
```

Save my state for the given plan over the given interval for the given set of resources. If any parameters are NULL I will save all that I contain for that parameter.

- saveStateLists

```
virtual int saveStateLists(const char* plan, const HEpochInterval& interval, const HVList* rsIds);
```

Save the resource states given the resources, plan and time interval. If no resources are specified, all resources are saved. If no plan is specified, all plans are saved, if no interval is set, the entire time span is saved.

- srm

```
virtual PlRpRmSrm* srm() const;
```

Get my resource model

- srm

```
virtual int srm(PlRpRmSrm* anSrm);
```

Set my resource model

6.3.21.3.1 PIRpRmCatalog Class

Overview:

An instance of PIRpRmCatalog is owned by a PIRpRmSrm client. PIRpRmCatalog adds additional behaviour to the SrmCatalog to handle registration and notification specific to PIRpRmSrm clients.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PIRpRmCatalog();
```

Default constructor

```
PIRpRmCatalog(const PIRpRmCatalog& orig);
```

Copy constructor

```
virtual ~PIRpRmCatalog();
```

Destructor

Operations:

- addClient

```
virtual int addClient(HAddress* cliAdr);
```

Add a client to me. I will clone the passed-in HAddress. overridden from base class to make sure I add an PIRpRmCliEntry.

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpRmCatalog );
```

An instance of PIRpRmCatalog is owned by a PIRpRmSrm client. PIRpRmCatalog adds additional behaviour to the SrmCatalog to handle registration and notification specific to PIRpRmSrm clients.

- operator =

```
PIRpRmCatalog& operator =(const PIRpRmCatalog& orig);
```

Assignment operator

- registerChgRs

```
virtual int registerChgRs(RResource* rs, HAddress* fromWhom);
```

Register changed resource from a client

- registerChgRs

```
virtual int registerChgRs(RResource* rs);
```

Register changed resource

- registerDelRs

```
virtual int registerDelRs(RResource* rs);
```

Register a deleted resource

- registerDelRs

```
virtual int registerDelRs(RResource* rs, HAddress* fromWhom);
```

Register a deleted resource from a client

- registerNewRs

```
virtual int registerNewRs(RResource* rs, HAddress* fromWhom);
```

Register new resource from a client

- registerNewRs

```
virtual int registerNewRs(RResource* rs);
```

Register new resource

- registerPlanNew

```
virtual int registerPlanNew(PlPlPlan* plan);
```

Register new plan. Send out a notice to any clients interested in plan names.

- registerPlanNew

```
virtual int registerPlanNew(PlPlPlan* plan, HAddress* fromWhom);
```

Register new plan.

- registerPlanNew

```
virtual int registerPlanNew(const char* planName);
```

Register new plan. Defer to base class. The following function had to be overridden to avoid hiding it, since the function has been overloaded in this class.

- registerPlanNew

```
virtual int registerPlanNew(const char* planName, HAddress* fromWhom);
```

Register new plan. Defer to base class. The following function had to be overridden to avoid hiding it, since the function has been overloaded in this class.

6.3.21.3.2 PIRpRmSigHand Class

Overview:

Instances of PIRpRmSigHand are signal catchers for a resource model to detect termination signals and save our resource model states.

Export Control: Public

Inheritance Relationships:

Attributes:

`myRm: PIRpRmAppl*`

The resource model application class who I contact when I receive a signal

Constructors and Destructor:

`PIRpRmSigHand();`

Default constructor

`PIRpRmSigHand(const PIRpRmSigHand& orig);`

Copy constructor

`virtual ~PIRpRmSigHand();`

Destructor

Operations:

- `DECLARE_LOCAL_CLASS`

`int DECLARE_LOCAL_CLASS(PIRpRmSigHand);`

Instances of PIRpRmSigHand are signal catchers for a resource model to detect termination signals and save our resource model states.

- `operator =`

`PIRpRmSigHand& operator =(const PIRpRmSigHand& orig);`

Assignment operator

- `rm`

`virtual PIRpRmAppl* rm() const;`

Get my ecs resource model

- rm

```
virtual int rm(PIRpRmAppl* anRm);
```

Set my ecs resource model

- terminate

```
void terminate(int );
```

Signals - terminate

- usr1Signal

```
virtual void usr1Signal(int );
```

usr signal1, override to ask my resource model to save its state

6.3.21.3.3 PIRpRmSrmNoid Class

Overview:

The PIRpRmSrmNoid is an ipc msgAgent that services resource model requests. These requests include allocation of activities, adding and removing resources, and requesting information about the resources.

Additional message handling behavior for ecs-specific messages is added including handling of oversubscription, plans, etc.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PIRpRmSrmNoid();
```

Default constructor.

```
PIRpRmSrmNoid(const PIRpRmSrmNoid& noid);
```

Copy constructor, restricted

```
virtual ~PIRpRmSrmNoid();
```

Destructor.

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpRmSrmNoid );
```

The PIRpRmSrmNoid is an ipc msgAgent that services resource model requests. These requests include allocation of activities, adding and removing resources, and requesting information about the resources.

Additional message handling behavior for ecs-specific messages is added including handling of oversubscription, plans, etc.

- establishInterest

```
virtual int establishInterest();
```

Establish interest with my agent in all resource model messages

- handleActPoolGet

```
virtual int handleActPoolGet(HMsgConn* curCon, HAddress* curAdr,  
PIRpClMsgGetActPool* curMsg);
```

Handle a message which requests a copy of the current activity pool.

- handleAllocAct

```
virtual int handleAllocAct(HMsgConn* curCon, HAddress* curAdr,  
SMsgAllocAct* curMsg);
```

Handle a message to do an allocation with constraint checking and a supplied RActivity. Overridden to see if we have the derived message type that knows about oversubscription of messages. If so, we set the Srm's derived ecs SaAllocator accordingly.

- handleAllocCheck

```
virtual int handleAllocCheck(HMsgConn* conn, HAddress* adr,  
SMsgAllocCheck* msg);
```

Handle an inbound message instructing me to check an allocation. Overridden to see if we have the derived message type that knows about oversubscription of messages. If so, we pass that information along to the PIRpRmSrm so that it can do the right thing.

- handleAllocMove

```
virtual int handleAllocMove(HMsgConn* conn, HAddress* adr,  
SMsgAllocMove* msg);
```

Handle an inbound message instructing me to move an allocation. Overridden to see if we have the derived message type that knows about oversubscription of messages. If so, we set the Srm's derived ecs SaAllocator accordingly.

- handleAllocRqst

```
virtual int handleAllocRqst(HMsgConn* curCon, HAddress* curAdr,  
SMsgAllocRqst* curMsg);
```

Handle a message to do an allocation with constraint checking, and a collection of RSchRqsts. Overridden to see if we have the derived message type that knows about oversubscription of messages. If so, we set the Srm's derived ecs SaAllocator accordingly.

- handleAlloc

```
virtual int handleAlloc(HMsgConn* conn, HAddress* adr,  
SMsgAlloc* msg);
```

Handle an inbound message instructing me to do an allocation. Overridden to see if we have the derived message type that knows about oversubscription of messages. If so, we set the Srm's derived ecs SaAllocator accordingly.

- handleChgRs

```
virtual int handleChgRs(HMsgConn* curCon, HAddress* curAdr,  
PlRpClMsgRsChg* curMsg);
```

Handle a message which requests a resource to be changed in the pool.

- handleDelRs

```
virtual int handleDelRs(HMsgConn* curCon, HAddress* curAdr,  
PlRpClMsgRsDel* curMsg);
```

Handle a message which requests a resource to be deleted from the pool.

- handleMessage

```
virtual int handleMessage(HMsgConn* curCon, HAddress* curAdr,  
HMessage* curMsg);
```

Handle incoming message.

- handleNewPlan

```
virtual int handleNewPlan(HMsgConn* curCon, HAddress* curAdr,  
PlRpClMsgPlanNew* curMsg);
```

Handle a message which requests plans. This is similar to my base SrmNoid::handlePlanNew

- handleNewRs

```
virtual int handleNewRs(HMsgConn* curCon, HAddress* curAdr,  
PlRpClMsgRsNew* curMsg);
```

Handle a message which requests a new resource to be added to the pool.

- handlePlanSave

```
virtual int handlePlanSave(HMsgConn* curCon, HAddress* curAdr,  
PlRpClMsgPlanSave* curMsg);
```

Handle a message which requests a plan to be saved to a file.

- `handlePlansGet`

```
virtual int handlePlansGet(HMsgConn* curCon, HAddress* curAdr,  
PIRpClMsgGetPlans* curMsg);
```

Handle a message which requests plans.

- `handleRsNms`

```
virtual int handleRsNms(HMsgConn* curCon, HAddress* curAdr,  
PIRpClMsgGetRsNms* curMsg);
```

Handle a message which requests a list of all the resource names that the srm knows about.

- `operator =`

```
PIRpRmSrmNoid& operator =(const PIRpRmSrmNoid& noid);
```

Restrict access to assignment.

- `srm`

```
virtual int srm(Srm* aSrm);
```

Set my visibility to an Srm. I do not assume ownership. Overridden from base class to ensure that I am actually associated with an PIRpRmSrm.

- `srm`

```
virtual Srm* srm() const;
```

Get my visibility to an Srm.

6.3.21.3.4 PIRpRmSrm Class

Overview:

Instances of PIRpRmSrm are resource model editors that know about ecs notions, plan accesses, etc.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myActivityIdFactory: PIRpRcActIdFact*
```

The activity id factory

myAllocIdFactory: PIRpRcAllocIdFact*

The allocation id factory

myAppl: PIRpRmAppl*

The parent application

myResourceIdFactory: PIRpRcResourceIdFact*

The resource id factory

Constructors and Destructor:

PIRpRmSrm();

Default constructor

PIRpRmSrm(const PIRpRmSrm& orig);

Copy constructor, restricted copy

virtual ~PIRpRmSrm();

Destructor

Operations:

- allocCheck

**virtual int allocCheck(const char* planName, const
HEpochInterval& intvl, int rsId, int actId, int ovrAllowed);**

Check allocation (do constraint-checking, but do not allocate). This determines if an allocation COULD be performed, allowing the caller to specify if oversubscription is permitted or not. The specified resource is checked to see if it is a PIRpScResource type, which knows about oversubscription issues. If so, then it is passed the oversubscription flag. If not, then the SResource checkAll is used.

- allocCheck

**virtual int allocCheck(const char* planName, const
HEpochInterval& intvl, int rsId, int actId);**

Check allocation (do constraint-checking, but do not allocate). This determines if an allocation COULD be performed. The default is to not permit oversubscription.

- appl

virtual const PIRpRmAppl* appl() const;

Get my parent application

- appl

```
virtual int appl(PIRpRmAppl* anAppl);
```

Set my parent application

- chgRs

```
virtual int chgRs(HAddress* fromAdr, RResource* chgRs, HVList*  
rsIds);
```

A client has sent a request to modify a resource I do not assume ownership to the passed in HAddress, which may be NULL.

- createCatalog

```
virtual int createCatalog();
```

Create and initialize my ECS client catalog

- createIdFactories

```
virtual int createIdFactories();
```

Create the id generator factories

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpRmSrm );
```

Instances of PIRpRmSrm are resource model editors that know about ecs notions, plan accesses, etc.

- delRs

```
virtual int delRs(HAddress* fromAdr, RResource* delRs, HVList*  
rsIds);
```

A client has sent a request to delete a resource I do not assume ownership to the passed in HAddress, which may be NULL.

- init

```
virtual int init();
```

Initialize. Overrides Srm::init() to initialize my ECS specifics

- newPlan

```
virtual int newPlan(HAddress* fromAdr, const char* planName,  
HVList* rsIds);
```

A client has sent a request to create a new plan given the name of the plan only. Overrides Srm::newPlan to not hide newPlan(). I do not assume ownership to the passed in HAddress, which may be NULL.

- newPlan

```
virtual int newPlan(HAddress* fromAdr, P1P1Plan* newPlan,  
HVList* rsIds);
```

A client has sent a request to create a new ECS derived plan I do not assume ownership to the passed in HAddress, which may be NULL. I will only create a new state list on those resources whose id is in the passed in list

- newRs

```
virtual int newRs(HAddress* fromAdr, RResource* newRs, HVList*  
rsIds);
```

A client has sent a request to create a new resource I do not assume ownership to the passed in HAddress, which may be NULL.

- operator =

```
PlRpRmSrm& operator =(const PlRpRmSrm& orig);
```

Assignment operator, restricted assignment

- savePlan

```
virtual int savePlan(const char* plan, const HEpochInterval&  
interval, HVList* rsIds);
```

Save a plan's states and allocations. Portions of the plan may be saved by specifying the resources to save on a specific plan over a specific interval of time. If the collection is empty all resource states and allocations associated with those states are saved. If the plan is not specified, all plans are saved. If the interval is not set than the entire time span is saved.

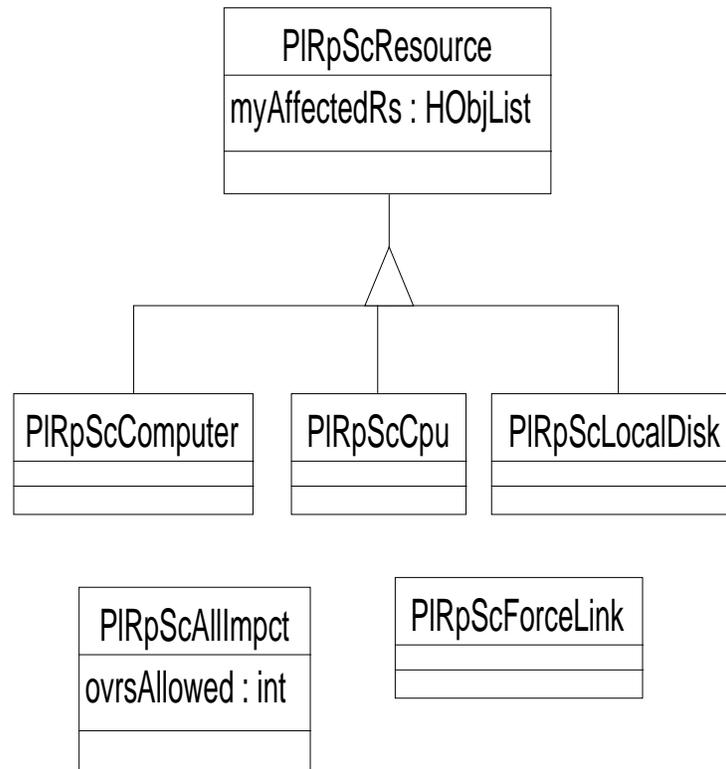
- setActId

```
virtual int setActId(RActivity* act);
```

Set activity id

6.3.22 Resource_Planning_Sc Class Category

6.3.22.1 Overview



6.3.22.2 Resource_Planning_Sc

6.3.22.3 PIRpScAllImpct

Overview:

class `PIRpScAllImpct` (Impact Scheduler)

Instances of `PIRpScAllImpct` add the knowledge of oversubscription to the impact scheduler also prevent any activity from being broken.

Export Control: Public

Inheritance Relationships:

Attributes:

`ovrsAllowed: int`

The oversubscription allowed flag

Constructors and Destructor:

`PlRpScAllImpct();`

Default constructor

`PlRpScAllImpct(const PlRpScAllImpct& orig);`

Copy constructor - restricted access

`virtual ~PlRpScAllImpct();`

Destructor

Operations:

- actBreakable

`virtual int actBreakable(RSimpleAct* act);`

Return TRUE if the act is breakable, FALSE if not. If the activity is a GenericAct, check its breakable flag overrides SaAllImpct::actBreakable

- allowOvrs

`virtual int allowOvrs() const;`

Get oversubscription allowed flag

- allowOvrs

`virtual void allowOvrs(int);`

Set oversubscription allowed flag

- checkAlloc

`virtual int checkAlloc(RSimpleAct* act, RSchOpp* opp);`

Return TRUE if the given activity can be allocated as specified in the given opportunity. If the resource involved is an PlRpScResource, the checkAlloc behavior with the oversubscription flag is called. overrides SaAllImpct::checkAlloc

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpScAllImpct );
```

```
class PIRpScAllImpct ( Impact Scheduler)
```

Instances of PIRpScAllImpct add the knowledge of oversubscription to the impact scheduler. They also prevent any activity from being broken.

- detBmpdActs

```
virtual int detBmpdActs(SResource* sRs, const HEpochInterval& reqIntvl, HObjCollection& bmpdActs);
```

Determine the activities to be bumped by asking the sRs, if it is an PIRpScResource, to do it itself. overrides SaAllImpct::detBmpdActs()

- operator =

```
PIRpScAllImpct& operator =(const PIRpScAllImpct& orig);
```

Assignment operator - restricted access

- whenAlloc

```
virtual int whenAlloc(RSimpleAct* act, RSchOpp* opp, HObjCollection& col);
```

Determine when the given activity can be allocated on the plan within the given interval, according to the specifications in the given opportunity. Return allowable intervals in col. If the resource involved is an PIRpScResource, the whenAlloc behavior with the oversubscription flag is called. overrides SaAllImpct::whenAlloc

6.3.22.3.1 PIRpScComputer Class

Overview:

Instances of PIRpScComputer are schedulable computer resources.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpScResource

Attributes:

Constructors and Destructor:

```
PIRpScComputer();
```

Default constructor

```
PIRpScComputer(const PIRpScComputer& orig);
```

Copy constructor

```
virtual ~PIRpScComputer();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpScComputer );
```

Instances of PIRpScComputer are schedulable computer resources.

- get

```
virtual int get(istream& istr);
```

Get object from a stream

- operator =

```
PIRpScComputer& operator =(const PIRpScComputer& orig);
```

Assignment operator

- put

```
virtual int put(ostream& ostr) const;
```

Put object onto a stream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get object from a XDR stream

6.3.22.3.2 PIRpScCpu Class

Overview:

Instances of PIRpScCpu are schedulable CPU resources.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpScResource

Attributes:

Constructors and Destructor:

```
PlRpScCpu();
```

Default constructor

```
PlRpScCpu(const PlRpScCpu& orig);
```

Copy constructor

```
virtual ~PlRpScCpu();
```

Destructor

Operations:

- addAllocNode

```
virtual int addAllocNode(RSimpleAct* act, const HEpochInterval&  
interval, const char* planName);
```

Add the appropriate alloc node for me to the given activity for the plan and interval.

- createRsState

```
RRsState* createRsState(RSimpleAct* act, const HEpochInterval&  
intvl);
```

Create the appropriate resource state for this activity and interval.

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpScCpu );
```

Instances of PlRpScCpu are schedulable CPU resources.

- get

```
virtual int get(istream& istr);
```

Get object from a stream

- operator =

```
PlRpScCpu& operator =(const PlRpScCpu& orig);
```

Assignment operator

- put

```
virtual int put(ostream& ostr) const;
```

Put object onto a stream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get object from XDR a stream

6.3.22.3.3 PIRpScForceLink Class

Overview:

class PIRpScForceLink (force link for schedule resource classes)

Instances of PIRpScForceLink are used to force linkage of schedule resource model classes which may not be present at link time.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PIRpScForceLink();
```

Default constructor

```
PIRpScForceLink(const PIRpScForceLink& orig);
```

Copy constructor

```
virtual ~PIRpScForceLink();
```

Destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpScForceLink );
```

class PIRpScForceLink (force link for schedule resource classes)

Instances of PIRpScForceLink are used to force linkage of schedule resource model classes which may not be present at link time.

- linkage

```
void linkage();
```

Protocol for forcing linkage of class objects not known at run time. Create dummy objects of all the types which you desire to force into the link. This behavior is never actually called at run time, just here to force the compiler to link classes in.

- operator =

```
PIRpScForceLink& operator =(const PIRpScForceLink& orig);
```

Assignment operator

6.3.22.3.4 PIRpScLocalDisk Class

Overview:

Instances of PIRpScLocalDisk are schedulable local disk resources.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpScResource

Attributes:

Constructors and Destructor:

```
PIRpScLocalDisk();
```

Default constructor

```
PIRpScLocalDisk(const PIRpScLocalDisk& orig);
```

Copy constructor

```
virtual ~PIRpScLocalDisk();
```

Destructor

Operations:

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpScLocalDisk );
```

Instances of PIRpScLocalDisk are schedulable local disk resources.

- get

```
virtual int get(istream& istr);
```

Get object from a stream

- operator =

```
PIRpScLocalDisk& operator =(const PIRpScLocalDisk& orig);
```

Assignment operator

- put

```
virtual int put(ostream& ostr) const;
```

Put object onto a stream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Set/get object from a XDR stream

6.3.22.3.5 PIRpScResource Class

Overview:

class PIRpScResource (base class schedulable resource for ECS)

This class sets up protocol for all ECS schedulable resources. It also adds a list of associated scheduling resources which will be notified when the state of this resource changes due to a scheduling request.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myAffectedRs: HObjList
```

List of affected resources

Constructors and Destructor:

```
PIRpScResource();
```

Constructor

```
PIRpScResource(const PIRpScResource& rsrc);
```

Copy constructor

```
virtual ~PIRpScResource();
```

Destructor

Operations:

- actAllowed

```
virtual int actAllowed(const RSimpleAct* act);
```

Return TRUE if given activity is allowed to be scheduled on this resource. Default behavior is to return TRUE if activity is a PIACActivity. Derived classes should override for specialized activities

- actStateType

```
virtual HClassDesc& actStateType();
```

Returns a class description for the type of state which I will add to my resource when I need to allocate an activity.

- addAllocNode

```
virtual int addAllocNode(RSimpleAct* act, const HEpochInterval& intvl, const char* plan);
```

Add the appropriate alloc node for me to the given activity for the plan and interval.

- addAssoc

```
virtual int addAssoc(HObject* obj);
```

Add a scheduling resource association Default behavior is to add to my list of affected resources. I retain visibility to the passed in resource, but do not assume ownership.

- alloc

```
virtual int alloc(RSimpleAct* act, const HEpochInterval& intvl, const char* plan, int ovrAllowed);
```

Try to allocate the activity to my resource for the given time interval. TRUE is returned if the allocation is successful. Tries the allocation on all of its associated schedulable resources, then calls checkAlloc and forceAll on my resource. This interface adds a flag indicating whether or not to allow oversubscription (ie. allow overlapping activities) of my resource. The name has been changed from all() to alloc() to prevent the need to override more than one behavior from this class.

- all

```
virtual int all(RSimpleAct* act, const HEpochInterval& intvl, const char* plan);
```

Original interface to do allocation in SResource. Overridden here to call alloc() above with ovrAllowed set to FALSE.

- checkAffectedRs

```
virtual int checkAffectedRs(RSimpleAct* act, const HEpochInterval& intvl, const char* plan, int ovrAllowed);
```

Check to see if the activity can be allocated on my affected resources for the given time interval. TRUE if allocation would be successful, FALSE otherwise.

- checkAlloc

```
virtual int checkAlloc(RSimpleAct* act, const HEpochInterval&
intvl, const char* plan, int ovrAllowed);
```

Check to see if the activity can be allocated to my resource for the given time interval. TRUE if allocation would be successful, FALSE otherwise. Checks to make sure that the activity can be allocated on my affected schedulable resources then checks for mutual exclusion if oversubscription is disallowed. This interface adds a flag indicating whether or not to allow oversubscription (ie. allow overlapping activities) of my resource. The name has been changed from checkAll() to checkAlloc() to prevent the need to override more than one behavior from this class in classes derived from this class.

- checkAll

```
virtual int checkAll(RSimpleAct* act, const HEpochInterval&
intvl, const char* plan);
```

original interface to check allocation in SResource. overridden here to call checkAlloc above with ovrAllowed set to FALSE.

- createRsState

```
virtual RRsState* createRsState(RSimpleAct* act, const
HEpochInterval& intvl);
```

Create the appropriate resource state for this activity and interval.

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpScResource );
```

class PIRpScResource (base class schedulable resource for ECS)

This class sets up protocol for all ECS schedulable resources. It also adds a list of associated scheduling resources which will be notified when the state of this resource changes due to a scheduling request.

- findActsToBump

```
virtual int findActsToBump(const char* plan, const
HEpochInterval& intvl, HObjCollection& col, int ovrAllowed);
```

Find the activities allocated to me that would need to be bumped in order to make room on me over the given interval on the given plan. Activities found are added to the collection, if they are not already in the collection. The number added is returned. Base class behavior is to add nothing to collection and return 0.

- forceAffectedRs

```
virtual int forceAffectedRs(RSimpleAct* act, const
HEpochInterval& intvl, const char* plan);
```

Force an allocation for the given activity onto my affected resources over the given time interval. TRUE if allocation is successful, FALSE otherwise.

- forceAll

```
virtual int forceAll(RSimpleAct* act, const HEpochInterval& intvl, const char* planName);
```

Create resource state and put it on my associated resource. This member creates an instance of RActState for the resource and tells the activity to remember the allocation. The exact state type added to the resource is determined by actStateType(), which may be overridden in derived classes. TRUE is returned if everything worked, FALSE otherwise.

- intersectList

```
virtual int intersectList(const HEpochIntervalList& list1, const HEpochIntervalList& list2, HObjCollection& retColl, int emptyRetColl);
```

Determine intersection of two lists and fill collection with results. If emptyRetColl is TRUE, retColl will be emptied before being filled with the intersection intervals. this enables you to specify the same list as both an input list and the output list if you wish, since the algorithm will completely process the input lists before clearing the output list. this technique can be useful if you are trying to compute the overall intersection of list1, list2, list3, and list4, for example, and you don't need to keep the original list1 around. then you could compute the desired intersection by making the calls: intersectList(list1, list2, list1, TRUE); intersectList(list1, list3, list1, TRUE); intersectList(list1, list4, list1, TRUE); if emptyRetColl is FALSE (the default), retColl will NOT be emptied before being filled with the intersection intervals.

this algorithm is used in ECS scheduling

- operator =

```
PlRpScResource& operator =(const PlRpScResource& rsrc);
```

Assignment operator

- printInterval

```
void printInterval(const HEpochInterval& intvl);
```

Print the given interval to stdout in a nice form. Used for debugging.

- remAffectedRs

```
virtual int remAffectedRs(RSimpleAct* act, const HEpochInterval& intvl, const char* plan);
```

Remove the allocation for the given activity from my affected resources over the given time interval. TRUE if removal is successful, FALSE otherwise.

- remAll

```
virtual int remAll(RSimpleAct* act, const HEpochInterval& intvl, const char* planName);
```

Remove the resource states generated by the activity over the given interval.

- whenAffectedRs

```
virtual int whenAffectedRs(RSimpleAct* act, const
HEpochInterval* intvl, const char* plan, HEpochIntervalList&
availList, int ovrAllowed);
```

Modify periods in availList to accommodate available periods in my affected resources.

- whenAlloc

```
virtual int whenAlloc(RSimpleAct* act, const HEpochInterval*
intvl, const char* plan, HObjCollection& col, int ovrAllowed);
```

Check to see when the activity could be allocated. The number of intervals added to the collection is returned. An interval is added to the collection if allocation is possible over that interval. This interface adds a flag indicating whether or not to allow oversubscription (ie. allow overlapping activities) of my resource. The name has been changed from whenAll() to whenAlloc() to prevent the need to override more than one behavior from this class.

- whenAll

```
virtual int whenAll(RSimpleAct* act, HEpochInterval* intvl,
const char* plan, HObjCollection& col);
```

Original interface to when allocation in SResource. Overridden here to call whenAlloc above with ovrAllowed set to FALSE.

6.3.23 Resource_Planning_Si Class Category

6.3.23.1 Overview

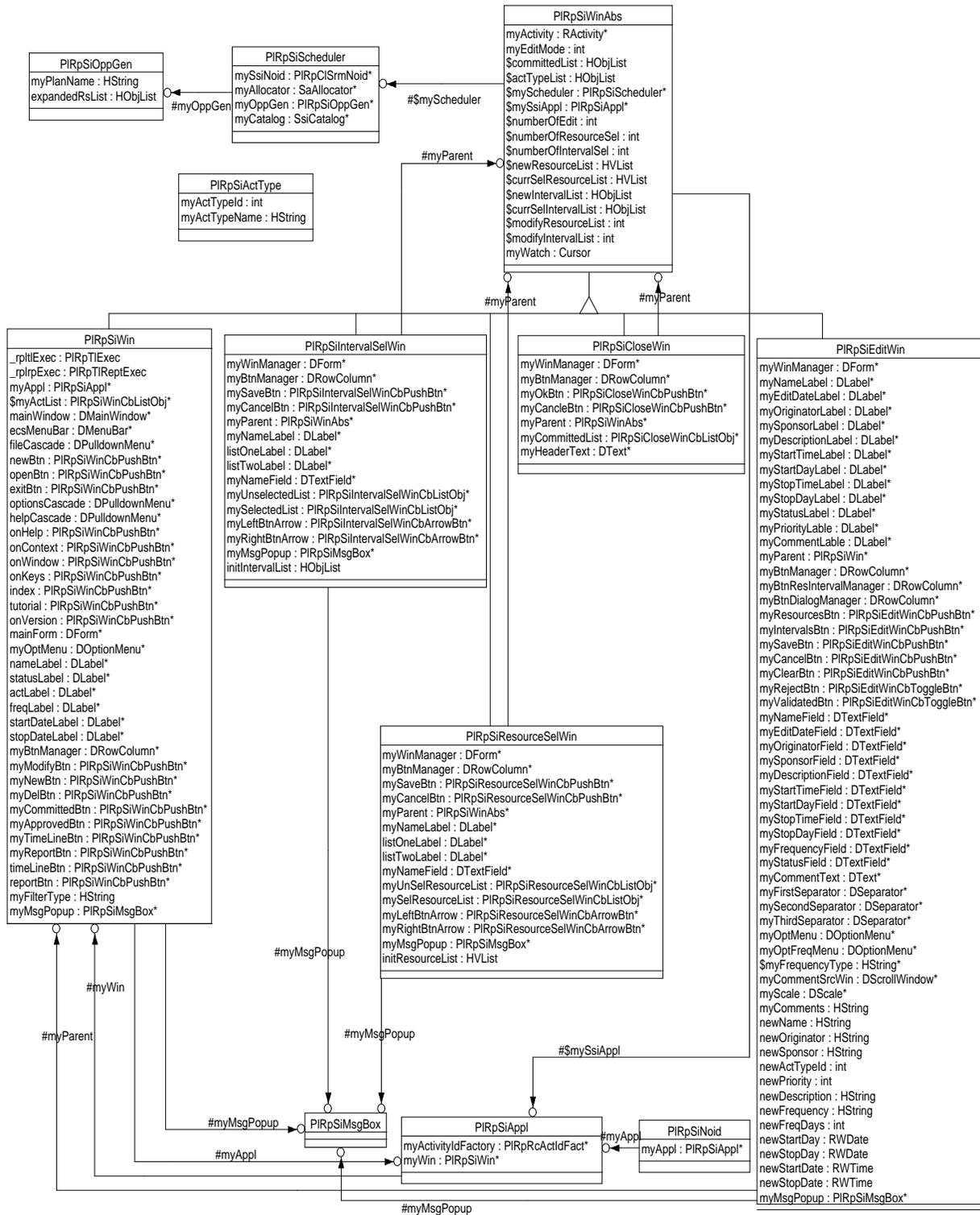


Figure 6.3.23.1-1 Resource_Planning_Si

6.3.23.2 Resource_Planning_Si Classes

6.3.23.3 PIRpSiActType Class

Overview:

PIRpSiActType is the same of one row in ACTIVITY_TYPE table. It has activity type id and activity name.

Export Control: Public

Inheritance Relationships:

Attributes:

`myActTypeId: int`

Activity type id

`myActTypeName: HString`

Activity type name

Constructors and Destructor:

`PIRpSiActType();`

Default constructor

`PIRpSiActType(const PIRpSiActType& orig);`

Copy constructor

`virtual ~PIRpSiActType();`

Destructor

Operations:

- actTypeId

`virtual int actTypeId() const;`

Get activity type id

- actTypeId

`virtual void actTypeId(int);`

Set activity type id

- actTypeName

```
virtual void actTypeName(const char* aActTypeName);
```

Set activity type name

- actTypeName

```
virtual const HString& actTypeName() const;
```

Get activity type name

- DECLARE_CLASS

```
int DECLARE_CLASS(PlRpSiActType );
```

PlRpSiActType is the same of one row in ACTIVITY_TYPE table. It has activity type id and activity name.

- get

```
virtual int get(istream& istr);
```

Get object from a stream

- operator =

```
PlRpSiActType& operator =(const PlRpSiActType& orig);
```

Assignment operator

- put

```
virtual int put(ostream& ostr) const;
```

Put object onto a stream

- xdr

```
virtual int xdr(XDR* xdrs);
```

Put/get object from a XDR stream

6.3.23.3.1 PlRpSiAppl Class

Overview:

Instance of PlRpSiAppl is derived from PlRpDiAppl.

The PlRpSiAppl is the System scheduling interface tool. This tool is the application frame work for providing generic scheduling.

Export Control: Public

Inheritance Relationships:

Inherits from PlRpDiAppl

Attributes:

`myActivityIdFactory: PlRpRcActIdFact*`

My activity id factory

`myWin: PlRpSiWin*`

Set my window

Constructors and Destructor:

`PlRpSiAppl(const PlRpSiAppl& orig);`

Copy constructor

`PlRpSiAppl();`

Default Constructor

`virtual ~PlRpSiAppl();`

Default Destructor

Operations:

- agent

`PlIpAgent* agent();`

The agent

- allocRemove

`virtual int allocRemove(const char* planName, const
HEpochInterval& when, int rsId, int actId);`

Remove an allocation from a scheduling resource

- cleanup

`virtual void cleanup();`

Cleanup

- connectionMade

`virtual int connectionMade();`

Get resources and plan name from resource model

- createEnviron

`virtual int createEnviron();`

Load reservations from the database

- createIdFactories

```
virtual int createIdFactories();
```

Init factory Id with the biggest number in the pool plus one. New reservation will use this id

- createMshNoid

```
virtual int createMshNoid();
```

Create a connection to the message handler.

- createSchedulableResources

```
virtual int createSchedulableResources();
```

Create SResources from RResources when finish getting RResources from resource model.

- create

```
virtual int create(const char* appClass, int& argc, char** argv,  
XrmOptionDescList options, Cardinal numOptions);
```

Create application.

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpSiAppl );
```

Instance of PIRpSiAppl is derived from PIRpDiAppl.

The PIRpSiAppl is the System scheduling interface tool. This tool is the application frame work for providing generic scheduling.

- edit

```
virtual int edit(int actId);
```

Set editable

- initCommunications

```
virtual int initCommunications();
```

Initialize communications. Override PIRpCIAppl behavior to delay creation of SRM noid.

- init

```
virtual int init();
```

Initialize application.

- makeWindow

```
virtual DWindow* makeWindow();
```

Override base behavior to make application specific window.

- newSrmNoid

```
virtual PlRpClSrmNoid* newSrmNoid();
```

Create the resource planning SRM noid.

- noid

```
virtual PlRpClSrmNoid* noid() const;
```

Get the pointer of the noid for all GUIs to use to pass message to resource model.

- operator =

```
PlRpSiAppl& operator =(const PlRpSiAppl& orig);
```

Assignment operator

- preSchedule

```
virtual int preSchedule();
```

Schedule reservation with status is either approved or committed once when it is in the activity pool

- shutDown

```
virtual void shutDown();
```

Shut down application.

6.3.23.3.2 PIRpSiCloseWin Class

Overview:

class PIRpSiCloseWin (Close Application Window)

An instance of this class display reservation that has approved status. Select Ok push button for changing approved status to committed status and exit or select Cancel to return to Resource Planning window

Export Control: Public

Inheritance Relationships:

Inherits from PIRpSiWinAbs

Attributes:

```
myBtnManager: DRowColumn*
```

Row column for ok and cancel push button widgets.

```
myCancleBtn: PlRpSiCloseWinCbPushBtn*
```

Cancel push button widget.

myCommittedList: PlRpSiCloseWinCbListObj*

List of approved reservations.

myHeaderText: DText*

Exited instruction text field.

myOkBtn: PlRpSiCloseWinCbPushBtn*

Ok push button widget.

myParent: PlRpSiWinAbs*

Pointer to parent window

myWinManager: DForm*

Main form widget.

Constructors and Destructor:

PlRpSiCloseWin(PlRpSiWinAbs* parent);

Constructor with visibility to parent window.

virtual ~PlRpSiCloseWin();

Default Destructor.

Operations:

- buildCommittedList

virtual int buildCommittedList();

Build list of reservations that need to be changed to commit.

- buildRowCols

virtual int buildRowCols();

Build row column for ok and cancel buttons.

- buildText

virtual int buildText();

Build Instruction for application text field.

- buildWindow

virtual int buildWindow();

Build Window Manager.

- cancelBtnCb

```
virtual void cancelBtnCb(PIRpSiCloseWinCbPushBtn* btn, XtPointer  
callData);
```

Callbacks, Popdown and destroy window, when no longer needed and return to Resource Planning window

- cleanup

```
virtual void cleanup();
```

Override base class member to provide window specific cleanup needs.

- createDisplay

```
virtual int createDisplay();
```

create a Top-Level Shell That is connected to the application.

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpSiCloseWin );
```

class PIRpSiCloseWin (Close Application Window)

An instance of this class display reservation that has approved status. Select Ok push button for changing approved status to committed status and exit or select Cancel to return to Resource Planning window

- init

```
virtual int init();
```

Override initialization from base class to provide specific window setup.

- OkBtnCb

```
virtual void OkBtnCb(PIRpSiCloseWinCbPushBtn* btn, XtPointer  
callData);
```

Callbacks, change status from approved to committed and exit

- updateCommittedList

```
virtual int updateCommittedList();
```

Fill up display list.

6.3.23.3.3 PIRpSiEditWin Class

Overview:

class PIRpSiEditWin (Show Reservation Details Window)

An instance of this class provides a view of the details associated with the selected reservation resource.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpSiWinAbs

Attributes:

myBtnDialogManager: DRowColumn*

Row Column for rejected and validated push button widgets.

myBtnManager: DRowColumn*

Row Column for save, clear, and cancel push button widgets.

myBtnResIntervalManager: DRowColumn*

Row Column for resource selection and interval selection push button widgets.

myCancelBtn: PIRpSiEditWinCbPushBtn*

Cancel push button widgets.

myClearBtn: PIRpSiEditWinCbPushBtn*

Clear push button widgets.

myCommentLable: DLabel*

Reservation comment label widget.

myCommentSrcWin: DScrollWindow*

Comment scroll list.

myComments: HString

Collect information from Reservation Request Edit/Definition.

Comments from Reservation Request Edit/Definition.

myCommentText: DText*

Reservation comment text field widget.

myDescriptionField: DTextField*

Reservation description text field widget.

myDescriptionLabel: DLabel*

Reservation description label widget.

myEditDateField: DTextField*

Reservation edit date text field widget.

myEditDateLabel: DLabel*
Reservation edit date label widget.

myFirstSeparator: DSeparator*
First separator.

myFrequencyField: DTextField*
Reservation frequency text field widget.

myFrequencyType: HString*
List of frequency types.

myIntervalsBtn: PlRpSiEditWinCbPushBtn*
Interval selection push button widgets.

myMsgPopup: PlRpSiMsgBox*
Pointer to message box popup

myNameField: DTextField*
Reservation name text field widget.

myNameLabel: DLabel*
Reservation name label widget.

myOptFreqMenu: DOptionMenu*
Option menu for frequency types.

myOptMenu: DOptionMenu*
Option menu for activity types.

myOriginatorField: DTextField*
Reservation originator text field widget.

myOriginatorLabel: DLabel*
Reservation originator label widget.

myParent: PlRpSiWin*
Pointer to parent window

myPriorityLable: DLabel*
Reservation priority label widget.

myRejectBtn: PlRpSiEditWinCbToggleBtn*
Rejected push button widgets.

myResourcesBtn: PlRpSiEditWinCbPushBtn*

Resource selection push button widgets.

mySaveBtn: PlRpSiEditWinCbPushBtn*

Save push button widgets.

myScale: DScale*

Priority widget.

mySecondSeparator: DSeparator*

Second separator.

mySponsorField: DTextField*

Reservation sponsor text field widget.

mySponsorLabel: DLabel*

Reservation sponsor label widget.

myStartDayField: DTextField*

Reservation startDay text field widget.

myStartDayLabel: DLabel*

Reservation start day label widget.

myStartTimeField: DTextField*

Reservation startTim text field widget.

myStartTimeLabel: DLabel*

Reservation start time label widget.

myStatusField: DTextField*

Reservation status text field widget.

myStatusLabel: DLabel*

Reservation status label widget.

myStopDayField: DTextField*

Reservation stopDay text field widget.

myStopDayLabel: DLabel*

Reservation stop day label widget.

myStopTimeField: DTextField*

Reservation stopTime text field widget.

myStopTimeLabel: DLabel*

Reservation stop time label widget.

myThirdSeparator: DSeparator*

Third separator.

myValidatedBtn: PIRpSiEditWinCbToggleBtn*

Validated push button widgets.

myWinManager: DForm*

Main form widget.

newActTypeId: int

ActTypeId from Reservation Request Edit/Definition.

newDescription: HString

Description from Reservation Request Edit/Definition.

newFreqDays: int

FreqDays from Reservation Request Edit/Definition.

newFrequency: HString

Frequency from Reservation Request Edit/Definition.

newName: HString

Name from Reservation Request Edit/Definition.

newOriginator: HString

Originator from Reservation Request Edit/Definition.

newPriority: int

Priority from Reservation Request Edit/Definition.

newSponsor: HString

Sponsor from Reservation Request Edit/Definition.

newStartDate: RWTime

StartDate from Reservation Request Edit/Definition.

newStartDay: RWDate

StartDay from Reservation Request Edit/Definition.

newStopDate: RWTime

StopDate from Reservation Request Edit/Definition.

newStopDay: RWDate

StopDay from Reservation Request Edit/Definition.

Constructors and Destructor:

```
PlRpSiEditWin(PlRpSiWin* parent);
```

Constructor with visibility to parent window.

```
virtual ~PlRpSiEditWin();
```

Default Destructor.

Operations:

- buildLabels

```
virtual int buildLabels();
```

Build labels widgets.

- buildOptMenus

```
virtual int buildOptMenus();
```

Build Option Menus for Activity Type selections and Frequency selections

- buildRowCols

```
virtual int buildRowCols();
```

Build row column for all push buttons widgets.

- buildScale

```
virtual int buildScale();
```

Build Scale for reservation's priority, default is from zero to 100

- buildScrText

```
virtual int buildScrText();
```

Build scroll text comments.

- buildSeparator

```
virtual int buildSeparator();
```

Build separators.

- buildTextFields

```
virtual int buildTextFields();
```

Build text fields.

- buildWindow

```
virtual int buildWindow();
```

Build Window Manager.
- cancelBtnCb

```
virtual void cancelBtnCb(PlRpsiEditWinCbPushBtn* btn, XtPointer callData);
```

Callbacks, Popdown and destroy window, when no longer needed.
- cleanupSelections

```
virtual void cleanupSelections();
```

Reset modifyResourceList and modifyIntervalList to FALSE; Empty newResourceList and newIntervalList.
- cleanup

```
virtual void cleanup();
```

Override base class member to provide window specific cleanup needs.
- clearBtnCb

```
virtual void clearBtnCb(PlRpsiEditWinCbPushBtn* btn, XtPointer callData);
```

Callbacks, reset all fields in the GUI with reservation's attributes.
- closeSelection

```
virtual int closeSelection();
```

Resource selection window and Interval selection window must be closed.
- collectEditDef

```
virtual int collectEditDef();
```

Get information from Reservation Request Edit/Definition.
- createDisplay

```
virtual int createDisplay();
```

Create a Top-Level Shell That is connected to the application.
- createIntervalSelWin

```
virtual PlRpsiIntervalSelWin* createIntervalSelWin();
```

Create interval selections window.
- createMsgBox

```
virtual int createMsgBox();
```

Create message box popup

- createNewResv

```
virtual int createNewResv();
```

Create new reservation.

- createResourceSelWin

```
virtual PIRpSiResourceSelWin* createResourceSelWin();
```

Create resource selections window.

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpSiEditWin );
```

class PIRpSiEditWin (Show Reservation Details Window)

An instance of this class provides a view of the details associated with the selected reservation resource.

- fillActTypeList

```
virtual int fillActTypeList();
```

Fill activity type options.

- fillCurrentTime

```
virtual void fillCurrentTime();
```

Use PTimeService get time for new reservation

- fillEditDef

```
virtual int fillEditDef();
```

Fill Resource Reservation Request Edit/Definition with reservation's attributes.

- fillFreqTypeList

```
virtual int fillFreqTypeList();
```

Fill frequency type options.

- getComments

```
virtual int getComments();
```

Fill comments scroll text with reservation's comments.

- getCurrentTime

```
virtual HDateTime& getCurrentTime(HDateTime& currentTime);
```

Get time from PTimeService and convert it to HDateTime.

- init

```
virtual int init();
```

Override initialization from base class to provide specific window setup.

- intervalsBtnCb

```
virtual void intervalsBtnCb(PlRpsIEditWinCbPushButton* btn,  
XtPointer callData);
```

Callbacks, pop up interval selections window.

- modifyDef

```
virtual int modifyDef();
```

Insert changes from Reservation Request Edit/Definition, re-populate interval list base on start date, stop date, and frequency.

- modifyResv

```
virtual int modifyResv();
```

Modify reservation.

- modifySelLists

```
virtual int modifySelLists();
```

Insert changes in resource list or interval list of selected reservation.

- modifyStatus

```
virtual int modifyStatus();
```

Insert change of status from new to rejected or validated for selected reservation.

- rejectBtnCb

```
virtual void rejectBtnCb(PlRpsIEditWinCbToggleBtn* , XtPointer  
);
```

Callbacks, set reservation's status from new to rejected.

- resourcesBtnCb

```
virtual void resourcesBtnCb(PlRpsIEditWinCbPushButton* btn,  
XtPointer callData);
```

Callbacks, pop up resource selections window.

- saveBtnCb

```
virtual void saveBtnCb(PlRpSiEditWinCbPushBtn* btn, XtPointer  
callData);
```

Callbacks, Check for changes in existing reservation or add new reservation in the activity pool.

- saveComments

```
virtual int saveComments();
```

Save comments scroll text into database.

- scanForDiffInDef

```
virtual int scanForDiffInDef();
```

Return zero for the same in reservation. Return one for different in reservation. Return two if it is bad like trying to re-use the same name with some other reservations.

- sendModifyResv

```
virtual int sendModifyResv();
```

Insert new changes into selected reservation.

- sendNewResv

```
virtual int sendNewResv();
```

Send new Reservation to resource model.

- tryToPopulateInterval

```
virtual int tryToPopulateInterval(PlRpAcResourceReservation*  
act);
```

Check any interval can be produced from inserted start date, stop date, and frequency.

- validatedBtnCb

```
virtual void validatedBtnCb(PlRpSiEditWinCbToggleBtn* ,  
XtPointer );
```

Callbacks, set reservation's status from new to validated.

6.3.23.3.4 PIRpSiIntervalSelWin Class

Overview:

class PIRpSiIntervalSelWin (Show Reservation's unSelected and Selected intervals windows)

An instance of this class provides a view of the details about unSelected and Selected intervals associated with the selected reservation.

Export Control: Public

Inheritance Relationships:

Inherits from `PIRpSiWinAbs`

Attributes:

`initIntervalList: HObjList`

NCR: ECSed05000 Initial list of intervals

`listOneLabel: DLabel*`

Unselected interval list label widget.

`listTwoLabel: DLabel*`

Selected interval list label widget.

`myBtnManager: DRowColumn*`

Row column for ok and cancel push button widgets.

`myCancelBtn: PIRpSiIntervalSelWinCbPushBtn*`

Cancel push button widget.

`myLeftBtnArrow: PIRpSiIntervalSelWinCbArrowBtn*`

Left arrow push button widget.

`myMsgPopup: PIRpSiMsgBox*`

Pointer to message box popup

`myNameField: DTextField*`

Reservation's name text field.

`myNameLabel: DLabel*`

Reservation's name label widget.

`myParent: PIRpSiWinAbs*`

Top-Level Shell.

`myRightBtnArrow: PIRpSiIntervalSelWinCbArrowBtn*`

Right arrow push button widget.

`mySaveBtn: PIRpSiIntervalSelWinCbPushBtn*`

Ok push button widget.

`mySelectedList: PIRpSiIntervalSelWinCbListObj*`

Selected interval scroll list

`myUnselectedList: PIRpSiIntervalSelWinCbListObj*`

Unselected interval scroll list

```
myWinManager: DForm*
```

Main form widget.

Constructors and Destructor:

```
PlRpSiIntervalSelWin(PlRpSiWinAbs* parent);
```

Constructor with visibility to parent window.

```
virtual ~PlRpSiIntervalSelWin();
```

Default Destructor.

Operations:

- buildArrowBtns

```
virtual int buildArrowBtns();
```

Build arrow button widgets.

- buildLabels

```
virtual int buildLabels();
```

Build labels widgets.

- buildLists

```
virtual int buildLists();
```

Build unSelected and Selected interval lists.

- buildRowCols

```
virtual int buildRowCols();
```

Build row column for ok and cancel buttons.

- buildTextField

```
virtual int buildTextField();
```

Build reservation's name text field.

- buildWindow

```
virtual int buildWindow();
```

Build Window Manager.

- cancelBtnCb

```
virtual void cancelBtnCb(PlRpSiIntervalSelWinCbPushBtn* btn,  
XtPointer callData);
```

Callbacks, Popdown and destroy window, when no longer needed.

- cleanup

```
virtual void cleanup();
```

Override base class member to provide window specific cleanup needs.

- createDisplay

```
virtual int createDisplay();
```

create a Top-Level Shell That is connected to the application.

- createMsgBox

```
virtual int createMsgBox();
```

Create message box popup

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpSiIntervalSelWin );
```

class PIRpSiIntervalSelWin (Show Reservation's unSelected and Selected intervals windows)

An instance of this class provides a view of the details about unSelected and Selected intervals associated with the selected reservation.

- fillLists

```
virtual int fillLists();
```

Fill unSelected and Selected lists.

- fillNameText

```
virtual int fillNameText();
```

Fill Reservation's name.

- init

```
virtual int init();
```

Override initialization from base class to provide specific window setup.

- leftArrBtnCb

```
virtual void leftArrBtnCb(PlRpSiIntervalSelWinCbArrowBtn* btn,  
XtPointer callData);
```

Callbacks, Move from Selected list to unSelected list.

- moveSelections

```
virtual void moveSelections(PlRpSiIntervalSelWinCbListObj*
fromList);
```

Set myOnOff from TRUE to FALSE, or from FALSE to TRUE.

- rightArrBtnCb

```
virtual void rightArrBtnCb(PlRpSiIntervalSelWinCbArrowBtn* btn,
XtPointer callData);
```

Callbacks, Move from unSelected list to Selected list.

- saveBtnCb

```
virtual void saveBtnCb(PlRpSiIntervalSelWinCbPushBtn* btn,
XtPointer callData);
```

Callbacks, Check for changes in Reservation's interval list.

6.3.23.3.5 PIRpSiMsgBox Class

Overview:

Class PIRpSiMsgBox (Resource Reservation Editor Message Popup) An instance of this provides a message popup box for displaying information to the user

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PlRpSiMsgBox();
```

Constructor.

```
virtual ~PlRpSiMsgBox();
```

Default Destructor.

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpSiMsgBox );
```

Class PIRpSiMsgBox (Resource Reservation Editor Message Popup) An instance of this provides a message popup box for displaying information to the user

- init

```
virtual int init(DWidget* parent);
```

Initialize message box.

- show

```
virtual void show(const char* msg);
```

Popup message box with given message string.

6.3.23.3.6 PIRpSiNoid Class

Overview:

class PIRpSiNoid (derived PIRpClSrmNoid)

I'm a noid who is only interested in handling resource related SrmClient messages.

I'm the interface between the system resource planning and the srm.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpClSrmNoid

Attributes:

```
myAppl: PIRpSiAppl*
```

Pointer to Resource planning window.

Constructors and Destructor:

```
PIRpSiNoid(PIRpSiAppl* appl);
```

Constructor with application class.

```
PIRpSiNoid();
```

Constructor

```
PIRpSiNoid(const PIRpSiNoid& orig);
```

Copy constructor

```
virtual ~PIRpSiNoid();
```

Destructor

Operations:

- appl

```
virtual void appl(PIRpSiAppl* appl);
```

Give me visibility to my application.

- connEstablished

```
virtual int connEstablished();
```

When connection is established, initialize.

- DECLARE_CLASS

```
int DECLARE_CLASS(PIRpSiNoid );
```

class PIRpSiNoid (derived PIRpClSrmNoid)

I'm a noid who is only interested in handling resource related SrmClient messages.

I'm the interface between the system resource planning and the srm.

- handleAllocRem

```
virtual int handleAllocRem(HMsgConn* , HAddress* , SMsgAllocRem* );
```

Method to handle a message indicating whether an allocation remove that I requested was successful.

- handlePlanNames

```
virtual int handlePlanNames(HMsgConn* , HAddress* , SMsgPlanNmsRsp* msg);
```

handle plan names.

- handlePlanNew

```
virtual int handlePlanNew(HMsgConn* , HAddress* , SMsgPlanNew* );
```

Method to handle new plan name.

- handlePlanSnap

```
virtual int handlePlanSnap(HMsgConn* , HAddress* , SMsgPlanSnap* msg);
```

Override install behavior for plans.

- init

```
virtual int init();
```

Initialize the application.

- `installActChg`

```
virtual int installActChg(RActivity* act);
```

 Override install behavior for change reservation.
- `installActDel`

```
virtual int installActDel(int actId);
```

 Override install behavior for delete reservation from the activity pool.
- `installActNew`

```
virtual int installActNew(RActivity* act);
```

 Override install behavior for new reservation.
- `installAllocUpd`

```
virtual int installAllocUpd(RUpdAllocAbs* anUpdate);
```

 Override install behavior for allocations.
- `installPlanClear`

```
virtual int installPlanClear(const char* planName,  
HEpochInterval* intvl, HVList* rsIds);
```

 Override install behavior for clear plan.
- `installPlanDel`

```
virtual int installPlanDel(const char* planName);
```

 Override install behavior for delete plan.
- `installPlanNew`

```
virtual int installPlanNew(const char* planName, HVList* rsIds);
```

 Override install behavior for new plan.
- `operator =`

```
PlRpSiNoid& operator =(const PlRpSiNoid& orig);
```

 Assignment operator
- `tick`

```
virtual void tick();
```

 refresh resource planning window.

6.3.23.3.7 PIRpSiOppGen Class

Overview:

class PIRpSiOppGen

Instances of PIRpSiOppGen are derived from SsiOppGen. It checks for available time slot of resource in reservation's resource list.

Export Control: Public

Inheritance Relationships:

Attributes:

expandedRsList: HObjList

Resource list.

myPlanName: HString

The plan for which I am generating opportunities.

Constructors and Destructor:

```
PIRpSiOppGen(const PIRpSiOppGen& orig);
```

Copy constructor

```
PIRpSiOppGen();
```

default constructor

```
virtual ~PIRpSiOppGen();
```

default destructor

Operations:

- addList

```
virtual int addList(PIRpRcResource* res);
```

Add a PIRpRcResource into resource list.

- addList

```
virtual int addList(PIRpRcComputer* computer);
```

Add a PIRpRcComputer into resource list.

- addList

```
virtual int addList(PlRpRcString* string);
```

Add a PlRpRcString into resource list.

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpSiOppGen );
```

class PlRpSiOppGen

Instances of PlRpSiOppGen are derived from SsiOppGen. It checks for available time slot of resource in reservation's resource list.

- expandedList

```
virtual HObjIter* expandedList();
```

Return the list of resource list.

- expandedResource

```
virtual int expandedResource(PlRpAcResourceReservation* reservation);
```

Expand all Computers and local disk associate with the string resource.

- operator =

```
PlRpSiOppGen& operator =(const PlRpSiOppGen& orig);
```

Assignment operator

- opportunities

```
virtual int opportunities(RActivity* act, RSchRqst* req, PlRpAcResourceReservation* unRes);
```

Attempt to create a scheduling opportunity for this request from the activity information plus return list of unreservable resources, reservable resources, conflicted reservations and its intervals.

- opportunities

```
virtual int opportunities(RActivity* act, RSchRqst* req);
```

Attempt to create a scheduling opportunity for this request from the activity information

- planName

```
virtual HString& planName();
```

get the plan name

- planName

```
virtual void planName(const char* plan);
```

set the plan name

- testRRsStateList

```
virtual int testRRsStateList(PlRpRcResource* resource, const
HEpochInterval* intvl, const char* listName, HString resvName,
PlRpAcResourceReservation* unRes);
```

Check for available spot in the given resource, given interval. return conflicted reservation and conflicted interval.

6.3.23.3.8 PIRpSiResourceSelWin Class

Overview:

class PIRpSiResourceSelWin (Show Reservation's unSelected and Selected resources windows)

An instance of this class provides a view of the details about unSelected and Selected resources associated with the selected reservation.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpSiWinAbs

Attributes:

initResourceList: HVList

List of resources selected by user when window is displayed initially

listOneLabel: DLabel*

Unselected resource list label widget.

listTwoLabel: DLabel*

Selected resource list label widget.

myBtnManager: DRowColumn*

Row column for ok and cancel push button widgets.

myCancelBtn: PlRpSiResourceSelWinCbPushBtn*

Cancel push button widget.

myLeftBtnArrow: PlRpSiResourceSelWinCbArrowBtn*

Left arrow push button widget.

myMsgPopup: PlRpSiMsgBox*

Pointer to message box popup

myNameField: DTextField*

Reservation's name text field.

myNameLabel: DLabel*

Reservation's name label widget.

myParent: PlRpSiWinAbs*

Pointer to parent window.

myRightBtnArrow: PlRpSiResourceSelWinCbArrowBtn*

Right arrow push button widget.

mySaveBtn: PlRpSiResourceSelWinCbPushBtn*

Ok push button widget.

mySelResourceList: PlRpSiResourceSelWinCbListObj*

Selected resourcescroll list.

myUnSelResourceList: PlRpSiResourceSelWinCbListObj*

Unselected resourcescroll list.

myWinManager: DForm*

Main form widget.

Constructors and Destructor:

```
PlRpSiResourceSelWin(PlRpSiWinAbs* parent);
```

Constructor with visibility to parent window.

```
virtual ~PlRpSiResourceSelWin();
```

Default Destructor.

Operations:

- buildArrowBtns

```
virtual int buildArrowBtns();
```

Build arrow button widgets.

- buildLabels

```
virtual int buildLabels();
```

Build labels widgets.

- buildLists

```
virtual int buildLists();
```

Build unSelected and Selected resource lists.

- buildRowCols

```
virtual int buildRowCols();
```

Build row column for ok and cancel buttons.

- buildTextField

```
virtual int buildTextField();
```

Build reservation's name text field.

- buildWindow

```
virtual int buildWindow();
```

Build Window Manager.

- cancelBtnCb

```
virtual void cancelBtnCb(PlRpSiResourceSelWinCbPushBtn* btn,  
XtPointer callData);
```

Callbacks, Popdown and destroy window, when no longer needed.

- cleanup

```
virtual void cleanup();
```

Override base class member to provide window specific cleanup needs.

- createDisplay

```
virtual int createDisplay();
```

Create a Top-Level Shell That is connected to the application.

- createMsgBox

```
virtual int createMsgBox();
```

Create message box popup

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpSiResourceSelWin );
```

class PlRpSiResourceSelWin (Show Reservation's unSelected and Selected resources windows)

An instance of this class provides a view of the details about unSelected and Selected resources associated with the selected reservation.

- fillLists

```
virtual int fillLists();
```

Fill unSelected and Selected lists.

- fillNameText

```
virtual int fillNameText();
```

Fill Reservation's name.

- init

```
virtual int init();
```

Override initialization from base class to provide specific window setup.

- leftArrBtnCb

```
virtual void leftArrBtnCb(PIRpSiResourceSelWinCbArrowBtn* btn,  
XtPointer callData);
```

Callbacks, Move from Selected list to unSelected list.

- moveSelections

```
virtual void moveSelections(PIRpSiResourceSelWinCbListObj*  
fromList, int addToNewList);
```

If addToNewList is TRUE, move from Unselected list to Selected list, Otherwise, move from Selected list to Unselected list

- rightArrBtnCb

```
virtual void rightArrBtnCb(PIRpSiResourceSelWinCbArrowBtn* btn,  
XtPointer callData);
```

Callbacks, Move from unSelected list to Selected list.

- saveBtnCb

```
virtual void saveBtnCb(PIRpSiResourceSelWinCbPushBtn* btn,  
XtPointer callData);
```

Callbacks, Check for changes in Reservation's resource list.

6.3.23.3.9 PIRpSiScheduler Class

Overview:

class PIRpSiScheduler

The PIRpSiScheduler is a generic Scheduling System user interface.

Export Control: Public

Inheritance Relationships:

Attributes:

`myAllocator: SaAllocator*`

Pointer to SaAllocator.

`myCatalog: SsiCatalog*`

Pointer to SsiCatalog

`myOppGen: PlRpSiOppGen*`

Pointer to PlRpSiOppGen, check for available slot.

`mySsiNoid: PlRpClSrmNoid*`

Pointer to noid.

Constructors and Destructor:

```
PlRpSiScheduler(const PlRpSiScheduler& orig);
```

Copy constructor

```
PlRpSiScheduler();
```

Default constructor

```
virtual ~PlRpSiScheduler();
```

Default destructor

Operations:

- allocator

```
virtual int allocator(SaAllocator* allocator);
```

Set my allocator explicitly.

- allocator

```
virtual SaAllocator* allocator();
```

Get my allocator explicitly.

- approved

```
virtual int approved(const char* planName, HObjCollection& col,  
HObjCollection& unAbleToScheList);
```

Change status from validated to approved or conflict, and send scheduled request to resource model.

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpSiScheduler );
```

class PIRpSiScheduler

The PIRpSiScheduler is a generic Scheduling System user interface.

- deleted

```
virtual int deleted(const char* planName,  
PIRpAcResourceReservation* reservation);
```

Change status to delete. if status is approved or committed, send remove Alloc to resource model.

- init

```
virtual int init();
```

Initialize PIRpSiSchedule

- modifyReservation

```
virtual int modifyReservation(const char* planName,  
PIRpAcResourceReservation* reservation);
```

Change status to new. if status is approved or committed, send remove Alloc to resource model.

- operator =

```
PIRpSiScheduler& operator =(const PIRpSiScheduler& orig);
```

Assignment operator

- unCommitted

```
virtual int unCommitted(const char* planName,  
PIRpAcResourceReservation* reservation);
```

Send remove Alloc to resource model.

6.3.23.3.10 PIRpSiWinAbs Class

Overview:

class PIRpSiWinAbs (Abstract Resource Planning)

an instance of this class shares static attributes for all windows.

Export Control: Public

Inheritance Relationships:

Attributes:

actTypeList: HObjList

Activity type list.

committedList: HObjList

List of approved reservations need to be changed to commit.

currSelIntervalList: HObjList

NCR: ECSed05000 List of currently selected intervals by the user

currSelResourceList: HVList

List of currently selected resources by the user

modifyIntervalList: int

ModifyIntervalList is FALSE if newIntervalList is the same with selected reservation's interval list. Otherwise, modifyIntervalList is TRUE.

modifyResourceList: int

ModifyResourceList is FALSE if newResourceList is the same with selected reservation's resource list. Otherwise, modifyResourceList is TRUE.

myActivity: RActivity*

My reservation.

myEditMode: int

My mode.

myScheduler: PlRpSiScheduler*

Pointer to my scheduler.

mySsiAppl: PlRpSiAppl*

Pointer to my PlRpSiAppl.

myWatch: Cursor

End of open one widget at the time Busy cursor

newIntervalList: HObjList

Interval list from Interval selection window.

newResourceList: HVList

Resource list from Resource selection window.

numberOfEdit: int

Begin of open one widget at the time Number of Resource reservation request edit/Definition window.

numberOfIntervalSel: int

Number of Interval selection window.

numberOfResourceSel: int

Number of Resource selection window.

Constructors and Destructor:

```
PLRpSiWinAbs(const PLRpSiWinAbs& orig);
```

Copy constructor

```
PLRpSiWinAbs();
```

Default constructor

```
virtual ~PLRpSiWinAbs();
```

Destructor

Operations:

- activity

```
virtual int activity(RActivity* rs);
```

Set my associated activity.

- activity

```
virtual RActivity* activity() const;
```

Get my associated activity.

- changeToCommit

```
virtual int changeToCommit();
```

Change status from approved to committed.

- cleanup

```
virtual void cleanup();
```

Delete window and tell PLRpSiAppl that I'm no longer active.

- clearDep

```
virtual int clearDep(HObject* obj);
```

Clear view dependency on the desired object. Default behavior is to clear dependency on my associated activity.

- DECLARE_ABS_CLASS

```
int DECLARE_ABS_CLASS(PlRpSiWinAbs );
```

class PlRpSiWinAbs (Abstract Resource Planning)

an instance of this class shares static attributes for all windows.

- edit_mode

```
virtual int edit_mode(enum PlRpSiWinAbs::EditMode eMode);
```

Set edit mode.

- edit_mode

```
virtual int edit_mode() const;
```

Get edit mode.

- findActTypeId

```
virtual int findActTypeId(HString actTypeName);
```

Return activity type id associate with activity type name.

- findActTypeName

```
virtual HString findActTypeName(int actTypeId);
```

Return activity type name associate with activity type id.

- getActTypeList

```
virtual int getActTypeList();
```

Get activity type list.

- init

```
virtual int init();
```

Initialize window. Derived classes will override this to provide application specific window setup.

- makeDep

```
virtual int makeDep(HObject* obj);
```

Make this view dependent on the desired object. Default behavior is to make window dependent on my associated activity.

- makeNewObject

```
virtual HObject* makeNewObject(const DpPrDbColValList&
thisColValList);
```

Make activity type object.

- makeObjects

```
virtual int makeObjects(const RWTValSlist& resultsList);
```

Make activity type list.

- notify

```
virtual int notify();
```

Notify PIRpSiAppl that I'm up and active.

- operator =

```
PIRpSiWinAbs& operator =(const PIRpSiWinAbs& orig);
```

Assignment operator

- selectedString

```
virtual HString* selectedString(DList* selList);
```

Return selected string in given scroll list.

- setWatch

```
virtual void setWatch(int flag);
```

Set my cursor to watch (if TRUE) or back to normal (if FALSE). overridden from base class to avoid core dump when there is no window.

- sissi

```
static PIRpSiAppl* sissi(PIRpSiAppl* appl);
```

Set the PIRpSiAppl application instance.

- sissi

```
static PIRpSiAppl* sissi();
```

Get the PIRpSiAppl application instance.

6.3.23.3.11 PIRpSiWin Class

Overview:

class PIRpSiWin (A derived DWindow)

This class is Motif Window for controlling the setup and display of the resource planning window.

Export Control: Public

Inheritance Relationships:

Inherits from `PIRpSiWinAbs`

Attributes:

actLabel: DLabel*

Activity type label widget.

ecsMenuBar: DMenuBar*

Main pull down menu bar.

exitBtn: PIRpSiWinCbPushBtn*

Exit button widget.

fileCascade: DPulldownMenu*

File pulldown menu widget

freqLabel: DLabel*

Frequency label widget.

helpCascade: DPulldownMenu*

Hepl pulldown menu widget

index: PIRpSiWinCbPushBtn*

Index push button widget.

mainForm: DForm*

Main form.

mainWindow: DMainWindow*

Main window.

myActList: PIRpSiWinCbListObj*

Pointer to display reservation list.

myAppl: PIRpSiAppl*

Pointer to PIRpSiAppl

myApprovedBtn: PIRpSiWinCbPushBtn*

Approved reservation push button widget.

myBtnManager: DRowColumn*

Row column for push button widget.

myCommittedBtn: PIRpSiWinCbPushBtn*

Committed reservation push button widget.

myDelBtn: PlRpSiWinCbPushBtn*

Deleted reservation push button widget.

myFilterType: HString

Current activity type in option menu.

myModifyBtn: PlRpSiWinCbPushBtn*

Modified reservation push button widget.

myMsgPopup: PlRpSiMsgBox*

Pointer to message box popup

myNewBtn: PlRpSiWinCbPushBtn*

New reservation push button widget.

myOptionsMenu: DOptionMenu*

Option push button widget.

myReportBtn: PlRpSiWinCbPushBtn*

Report push button widget.

myTimeLineBtn: PlRpSiWinCbPushBtn*

Time line push button widget.

nameLabel: DLabel*

Name label widget.

newBtn: PlRpSiWinCbPushBtn*

New button widget.

onContext: PlRpSiWinCbPushBtn*

On context push button widget.

onHelp: PlRpSiWinCbPushBtn*

On help push button widget.

onKeys: PlRpSiWinCbPushBtn*

On key push button widget.

onVersion: PlRpSiWinCbPushBtn*

On version push button widget.

onWindow: PlRpSiWinCbPushBtn*

On window push button widget.

openBtn: PlRpSiWinCbPushBtn*

Open button widget.

optionsCascade: DPullDownMenu*

Options pulldown menu widget

reportBtn: PlRpSiWinCbPushBtn*

Report push button widget from Option pull down bar.

startDateLabel: DLabel*

Start Date label widget.

statusLabel: DLabel*

Status label widget.

stopDateLabel: DLabel*

Stop Date label widget.

timeLineBtn: PlRpSiWinCbPushBtn*

Time line push button widget from Option pull down bar

tutorial: PlRpSiWinCbPushBtn*

Tutorial push button widget.

_rplrpExec: PlRpTlReptExec

PIRpTlReptExec -- the report GUI

_rpltlExec: PlRpTlExec

PIRpTlExec -- the rpltl (timeline) "spawner"

Constructors and Destructor:

```
PlRpSiWin(const PlRpSiWin& orig);
```

Copy constructor

```
PlRpSiWin(PlRpSiAppl* appl);
```

Default Constructor

```
virtual ~PlRpSiWin();
```

Default Destructor

Operations:

- actSelected

```
virtual int actSelected();
```

return true if select reservation.

- approvedBtnCb

```
virtual void approvedBtnCb(PlRpSiWinCbPushBtn* btn, XtPointer  
callData);
```

Attempt to approve reservation from validate. If it fails, status changes to conflict.

- buildActList

```
virtual int buildActList();
```

Build reservation list.

- buildButtons

```
virtual int buildButtons();
```

Build New, Modify, Approve, Commit, Time Line, Report push button widgets.

- buildLabels

```
virtual int buildLabels();
```

Build label widgets.

- buildMenuBar

```
virtual int buildMenuBar();
```

Build Menu bar.

- buildOptMenus

```
virtual int buildOptMenus();
```

Build activity type option menu.

- buildRowCols

```
virtual int buildRowCols();
```

Build row column for New, Modify, Approve, Commit, Time Line, Report push button widgets.

- committedBtnCb

```
virtual void committedBtnCb(PlRpSiWinCbPushBtn* btn, XtPointer  
callData);
```

Change status from approved to committed in display list.

- createCloseWin

```
virtual PlRpSiCloseWin* createCloseWin();
```

Create close window.

- createDisplay

```
virtual int createDisplay();
```

Overriden from base class to provide window display creation.

- createModWin

```
virtual PlRpSiEditWin* createModWin();
```

Create Resource Reservation request edit/Definition.

- createMsgBox

```
virtual int createMsgBox();
```

Create message box popup

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpSiWin );
```

class PlRpSiWin (A derived DWindow)

This class is Motif Window for controlling the setup and display of the resource planning window.

- delBtnCb

```
virtual void delBtnCb(PlRpSiWinCbPushBtn* , XtPointer cbs);
```

Change status to deleted for selected reservation and take it out from displayed list.

- exitBtnCb

```
virtual void exitBtnCb(PlRpSiWinCbPushBtn* , XtPointer cbs);
```

Exit application.

- fillOneResvStr

```
virtual HString& fillOneResvStr(HString& str, const  
PlRpAcResourceReservation* act);
```

Fill reservation string in displayed list.

- fillTypeList

```
virtual int fillTypeList();
```

Fill type option menu with each unique resource type found in the reservation.

- filterCb


```
virtual void filterCb(PlRpSiWinCbPushBtn* , XtPointer );
```

 Change filter type used for activity display.
- getCommittedList


```
virtual int getCommittedList();
```

 Collection of approved reservations.
- init


```
virtual int init();
```

 Initialize window.
- modifyBtnCb


```
virtual void modifyBtnCb(PlRpSiWinCbPushBtn* , XtPointer cbs);
```

 Modify selected reservation.
- newBtnCb


```
virtual void newBtnCb(PlRpSiWinCbPushBtn* , XtPointer cbs);
```

 Create a new reservation.
- operator =


```
PlRpSiWin& operator =(const PlRpSiWin& orig);
```

 Assignment operator
- padStr


```
virtual HString& padStr(HString& str, int len);
```

 Routine to right pad an HString with spaces
- parseCommandLine


```
virtual int parseCommandLine();
```

 Parse the command line.
- reportBtnCb


```
virtual void reportBtnCb(PlRpSiWinCbPushBtn* btn, XtPointer cbs);
```

 Display report window.
- selectedAct


```
virtual RActivity* selectedAct();
```

return selected reservation.

- setup

```
virtual int setup();
```

Setup display.

- timeLineBtnCb

```
virtual void timeLineBtnCb(PlRpSiWinCbPushBtn* btn, XtPointer  
cbs);
```

Display time line window.

- updateActList

```
virtual int updateActList();
```

Update the reservation display list from the reservation pool.

- update

```
virtual int update();
```

Update resource planning window.

6.3.24 Resource_Planning_TI Class Category

6.3.24.1 Overview

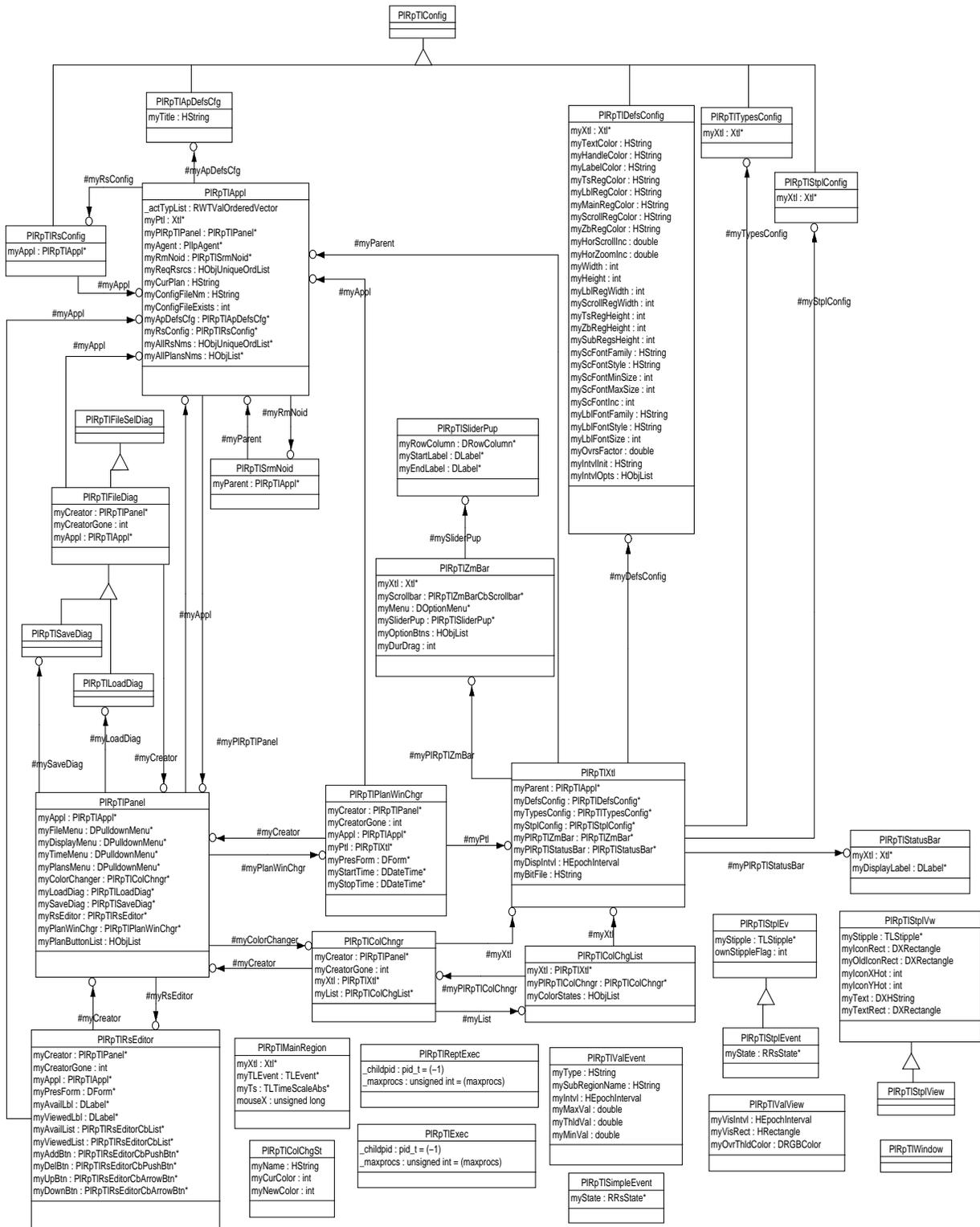


Figure 6.3.24.1-1 Resource_Planning_TI

6.3.24.2 Resource_Planning_TI Classes

6.3.24.3 PIRpTlApDefsCfg Class

Overview:

Instances of PIRpTlApDefsCfg handle configuration files for default display resource settings for the timeline.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpTlConfig

Attributes:

myTitle: HString

My frame title

Constructors and Destructor:

```
PIRpTlApDefsCfg();
```

constructor. calls base class constructor with default arguments.

```
PIRpTlApDefsCfg(const PIRpTlApDefsCfg& orig);
```

Copy - disabled

```
~PIRpTlApDefsCfg();
```

destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpTlApDefsCfg );
```

Instances of PIRpTlApDefsCfg handle configuration files for default display resource settings for the timeline.

- handleKeyword

```
virtual int handleKeyword(const HString& kywd);
```

Handle keyword. overrides ConfigurationFile::handleKeyword()

- operator =

```
PIRpTlApDefsCfg& operator =(const PIRpTlApDefsCfg& orig);
```

Assignment - disabled

- saveConfig

```
virtual int saveConfig(ostream& str);
```

Save the current set of resources to the given stream overrides PIRpTIConfig::saveConfig()

- title

```
virtual const HString& title() const;
```

Get my frame title

6.3.24.3.1 PIRpTIAppI Class

Overview:

PIRpTIAppI is the main application class for the timeline. It is responsible for setting up communication to an era as well as initialization of resources and their states. PIRpTIAppI also handles the cleanup and subsequent destroying of the appl.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myAgent: PIRpTIAgent*
```

my noid agent

```
myAllPlansNms: HObjList*
```

my list of the names of all available plans

```
myAllRsNms: HObjUniqueOrdList*
```

my list of the names of all available resources HObjCollection* myAllRsNms;

```
myApDefsCfg: PIRpTlApDefsCfg*
```

my PIRpTlXtl appl defaults configuration file

```
myConfigFileExists: int
```

flag indicating whether config file exists right now

```
myConfigFileNm: HString
```

my configuration file name

```
myCurPlan: HString
```

the current plan I am interested in

```
myPlRpTlPanel: PlRpTlPanel*
```

my PlRpTlPanel display

```
myPtl: Xtl*
```

my ECS timeline

```
myReqRsrcs: HObjUniqueOrdList
```

my requested resources, I own the list and am responsible for its deletion HObjList myReqRsrcs;

```
myRmNoid: PlRpTlSrmNoid*
```

my resource model noid

```
myRsConfig: PlRpTlRsConfig*
```

my PlRpTlXtl resource configuration file

```
_actTypList: RWTValOrderedVector
```

list of known activity types is fetched from the dbase:

Constructors and Destructor:

```
PlRpTlAppl();
```

default constructor

```
PlRpTlAppl(const PlRpTlAppl& orig);
```

copy disabled - nothing is copied

```
~PlRpTlAppl();
```

destructor

Operations:

- actTypNm

```
RWCstring actTypNm(int id);
```

fetch the reservation activity type name string by id

- allPlansNms

```
virtual HObjList* allPlansNms();
```

Set the list containing the names of all available plans.

- allPlansNms

```
virtual void allPlansNms(HObjList* );
```

Set the list containing the names of all available plans. I assume ownership upon set, DO NOT relinquish ownership upon get, and will delete the list.

- allRsNms

```
virtual HObjUniqueOrdList* allRsNms();
```

Get the list containing the names of all available resources. virtual HObjCollection* allRsNms();

- allRsNms

```
virtual void allRsNms(HObjCollection* );
```

Set the list containing the names of all available resources. I assume ownership upon set, DO NOT relinquish ownership upon get, and will delete the list

- changeViewedRs

```
virtual int changeViewedRs(const HObjCollection& newRsNms);
```

change the resources I am displaying to those whose names are given in the collection.

- cleanupForWinDestroy

```
virtual void cleanupForWinDestroy();
```

delete my PIRpTIXtl before the top level window gets destroyed so that all the regions will still be there when stuff cleans up

- cleanup

```
virtual void cleanup();
```

delete all dynamically created entities does my own, then calls base class behavior

- configFileExists

```
virtual int configFileExists();
```

get whether or not the config file exists right now

- configFileNm

```
virtual const HString& configFileNm() const;
```

get my config file name

- connectionMade

```
virtual int connectionMade();
```

upon notification that a connection to era has been made, proceed with rest of initialization, specifically display creation, resource model loading, and obtaining state

- createApDefsCfg

```
virtual PIRpTlApDefsCfg* createApDefsCfg();
```

create my appl defaults configuratin file

- createMshNoid

```
virtual int createMshNoid();
```

Create a connection to the message handler.

- createPtl

```
virtual xtl* createPtl();
```

create my PIRpTlXtl

- createRsConfig

```
virtual PlRpTlRsConfig* createRsConfig();
```

create my resource configuration file

- createSigMgr

```
virtual int createSigMgr();
```

Create my global signal manager

- currentPlan

```
virtual HString& currentPlan();
```

Get the current plan

- currentPlan

```
virtual void currentPlan(const char* );
```

Set the current plan

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlAppl );
```

PlRpTlAppl is the main application class for the timeline. It is responsible for setting up communication to an era as well as initialization of resources and their states. PIRpTlAppl also handles the cleanup and subsequent destroying of the appl.

- deletePlan

```
virtual int deletePlan();
```

delete plan events

- delRsStates

```
virtual int delRsStates(const HObjCollection& rsToDelete);
```

delete the state lists of resources I no longer want to display

- deregisterUpdates

```
virtual int deregisterUpdates();
```

deregister for all updates

- detConfigFile

```
virtual int detConfigFile();
```

determine the configuration file name. if specified on command line, the file must exist. if not specified on command line, use a default name which may or may not exist. return FALSE if file must exist, but does not

- detRsToAdd

```
virtual void detRsToAdd(const HObjCollection& newRSNms,  
HObjCollection& rsToAdd);
```

determine new resources not in the current set

- detRsToDel

```
virtual void detRsToDel(const HObjCollection& newRSNms,  
HObjCollection& rsToDelete);
```

determine current resources not in the new set

- displayWinAtRun

```
virtual void displayWinAtRun();
```

override to NOT map the window at run we don't want to map it until later, after we have gotten all our connections and got our data

- getInitialStates

```
virtual int getInitialStates();
```

get initial resource states

- getNewStates

```
virtual int getNewStates(const HObjCollection& rsToAdd);
```

request new state lists for new resources I now want to display

- initCommunications

```
virtual int initCommunications();
```

initialize message agent and rm noid and try to connect to era. this init is the first one called.

- initFromDb

```
virtual int initFromDb();
```

initialize the from database

- `init`

```
virtual int init();
```

initialize the application

- `loadActPool`

```
virtual int loadActPool();
```

create and load the activity pool with the activities associated with the sub regions

- `loadAllPlans`

```
virtual int loadAllPlans();
```

load the plans the SRM knows about.

- `loadAllRsrcNms`

```
virtual int loadAllRsrcNms();
```

load the names of all the resources the SRM knows about.

- `loadApplDefs`

```
virtual int loadApplDefs();
```

load my application defaults

- `loadConfig`

```
virtual int loadConfig(const char* filename);
```

load a new configuration from the given file and snap to that configuration

- `loadNewRs`

```
virtual int loadNewRs(const HObjCollection& rsToAdd);
```

load any new resources not already in the pool from the srm

- `loadResourcePool`

```
virtual int loadResourcePool();
```

create and load the resource pool with the resources associated with the sub regions

- `makeWindow`

```
virtual DWindow* makeWindow();
```

make the derived DWindow which holds the top-level shell

- `newRs`

```
virtual void newRs(RResource* aRs);
```

called when my noid gets notification of a new resource added to my pool.

- operator =

```
PlRpTlAppl& operator =(const PlRpTlAppl& orig);
```

assignment operator

- ptlChangeViewedRs

```
virtual int ptlChangeViewedRs(const HObjCollection& rsToDelete);
```

notify my PlRpTlXtl to change the resources it is displaying

- ptlSaveConfig

```
virtual int ptlSaveConfig(const char* filename);
```

tell my PlRpTlXtl to save its configuration to the given file

- reconfigPtl

```
virtual int reconfigPtl();
```

tell my PlRpTlXtl to reconfigure

- reconfigRs

```
virtual int reconfigRs();
```

reconfigure my resources. this includes deregistering for all current updates, clearing my resource pool, and requesting the new set of resources

- reconfigure

```
virtual int reconfigure();
```

reconfigure according to new config file

- registerForPlans

```
virtual int registerForPlans();
```

register for act, resource and plan instance updates

- registerForUpdates

```
virtual int registerForUpdates();
```

register for resource state updates

- reqRsrcs

```
virtual void reqRsrcs(const HObjUniqueOrdList& );
```

Set the resources

- reqRsrcs

```
virtual HObjUniqueOrdList& reqRsrcs();
```

Get the resources

- saveConfig

```
virtual int saveConfig(const char* filename);
```

save the current configuration to the given file

- setDefConfigFile

```
virtual void setDefConfigFile();
```

set the default config file name.

- snapToNewPlan

```
virtual int snapToNewPlan(const char* planName);
```

request snap to new plan

- srmCliNoid

```
virtual PIRpTlSrmNoid* srmCliNoid();
```

get my srm client noid

- stateUpdate

```
virtual void stateUpdate(RUpdState* aState);
```

called when my noid gets notification of a state update. I get the list of updates.

6.3.24.3.2 PIRpTlColChgList Class

Overview:

Instances of PIRpTlColChgList are lists which are part of the color mapper dialog.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myColorStates: HObjList
```

my list of color states

```
myPIRpTlColChngr: PIRpTlColChngr*
```

my parent color changer

```
myXtl: PlRpTlXtl*
```

the timeline

Constructors and Destructor:

```
PlRpTlColChgList(PlRpTlXtl* xtl, PlRpTlColChngr* chngr);
```

constructor. arguments are the xtl and my parent color changer

```
PlRpTlColChgList(const PlRpTlColChgList& orig);
```

copy - disabled

```
~PlRpTlColChgList();
```

destructor

Operations:

- apply

```
virtual void apply();
```

apply all the changes that have been made to the color mapper

- changeColor

```
virtual void changeColor(int ix, int cix);
```

change the color at the given index to the given color index (pixel)

- colorAt

```
virtual int colorAt(int ix);
```

return the color index at the given index in the list

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlColChgList );
```

Instances of PlRpTlColChgList are lists which are part of the color mapper dialog.

- deleteAllItems

```
virtual int deleteAllItems();
```

remove all items in the scrolling list. overridden to get rid of the objects that are associated with the list

- init

```
virtual int init();
```

init the scrolling list with all the types in TLTypes.

- operator =

```
PlRpTlColChgList& operator =(const PlRpTlColChgList& orig);
```

assignment - disabled

- reset

```
virtual void reset();
```

reset any states that might have changed to their original colors

- singleSelectionRef

```
virtual void singleSelectionRef(XtPointer callData, int row,  
void* ref);
```

called when a list item is selected

6.3.24.3.3 PlRpTlColChgSt Class

Overview:

Instances of PlRpTlColChgSt map colors to types.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myCurColor: int
```

my current color index

```
myName: HString
```

my type name

```
myNewColor: int
```

my new color index

Constructors and Destructor:

```
PlRpTlColChgSt(const char* typNm, int colorIndex);
```

constructor. arguments are my type name and my current color index.

```
PlRpTlColChgSt(const PlRpTlColChgSt& orig);
```

copy - disabled

```
~PlRpTlColChgSt();
```

destructor

Operations:

- colorChanged

```
virtual int colorChanged();
```

returns TRUE if curColor and newColor are different, FALSE otherwise.

- curColor

```
virtual void curColor(int index);
```

Set my current color index

- curColor

```
virtual int curColor() const;
```

Get my current color index

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlColChgSt );
```

Instances of PlRpTlColChgSt map colors to types.

- name

```
virtual void name(const char* typNm);
```

Set my type name

- name

```
virtual const HString& name() const;
```

Get my type name

- newColor

```
virtual void newColor(int index);
```

Set my new color index

- newColor

```
virtual int newColor() const;
```

Get my new color index

- operator =

```
PlRpTlColChgSt& operator =(const PlRpTlColChgSt& orig);  
assignment - disabled
```

- reset

```
virtual void reset();  
set the new color to be the current color.
```

- update

```
virtual void update();  
set the current color to be the new color.
```

6.3.24.3.4 PIRpTlColChngr Class

Overview:

Instances of PIRpTlColChngr can be used to specify colors for types.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myCreatorGone: int  
tells me if my creator is still around.
```

```
myCreator: PlRpTlPanel*  
the object that created me, if my creator is still around when I get destroyed, I will tell it I am gone.
```

```
myList: PlRpTlColChgList*  
My state list of types and colors
```

```
myXtl: PlRpTlXtl*  
the timeline
```

Constructors and Destructor:

```
PlRpTlColChngr(PlRpTlPanel* creator, PlRpTlXtl* xtl);  
constructor. argument is my creating PIRpTlPanel and the xtl
```

```
PlRpTlColChngr(const PlRpTlColChngr& orig);
```

copy - disabled

```
~PlRpTlColChngr();
```

destructor

Operations:

- applyChanges

```
virtual void applyChanges();
```

Apply all the color changes overrides DApplyDialog::applyChanges()

- colorChangerGone

```
virtual void colorChangerGone();
```

tell my parent that I have been destroyed. default behavior is to tell my ptl parent.

- colorChg

```
virtual void colorChg(DXColChgCbPushBtn* );
```

Called when the user selects a color in the palette. Update the state and preview color. overrides DXColChg::colorChg()

- createActionControls

```
int createActionControls();
```

Create my action controls. Overridden to add a DListObj to the action area. overrides DApplyDialog::createActionControls()

- creatorGone

```
virtual void creatorGone(int tf);
```

set whether or not my creator is gone. if my creator gets destroyed first, it will tell me.

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlColChngr );
```

Instances of PlRpTlColChngr can be used to specify colors for types.

- hide

```
virtual void hide();
```

Reset all colors, then do base class behavior overrides DPopup::hide()

- operator =

```
PlRpTlColChngr& operator =(const PlRpTlColChngr& orig);
```

assignment - disabled

- positionApply

```
void positionApply();
```

position my apply button. Override from base class to take account of my list, added to the action area. overrides DApplyDialog::positionApply()

- positionCancel

```
void positionCancel();
```

position my cancel button. Override from base class to take account of my list, added to the action area. overrides DApplyDialog::positionCancel()

- positionOk

```
void positionOk();
```

position my ok button. Override from base class to take account of my list, added to the action area. overrides DApplyDialog::positionOk()

- reconfigure

```
virtual int reconfigure();
```

reconfigure. empty the current scrolling list, then init it again to pick up a new set of types/colors.

- sizeActionForm

```
virtual int sizeActionForm();
```

Overridden to take into account my addition of my scroll list of region names. overrides DApplyDialog::sizeActionForm()

6.3.24.3.5 PIRpTIConfig Class

Overview:

PIRpTIConfig is the base class for all configuration files. It provides protocol for loading and saving configuration from the given file.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PlRpTlConfig();
```

constructor. calls base class constructor with default arguments.

```
PlRpTlConfig(const PlRpTlConfig& orig);
```

copy - disabled

```
~PlRpTlConfig();
```

destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlConfig );
```

PIRpTlConfig is the base class for all configuration files. It provides protocol for loading and saving configuration from the given file.

- getConfig

```
virtual int getConfig();
```

get any config info that I am storing which might have changed. protocol only.

- operator =

```
PlRpTlConfig& operator =(const PlRpTlConfig& orig);
```

assignment - disabled

- saveConfig

```
virtual int saveConfig(ostream& str);
```

save my current config info to the given stream. protocol only.

- save

```
virtual int save(const char* fn);
```

save. first get any config info which might have changed, then save the current values of my config info, along with the appropriate keywords to the given file. overrides ConfigurationFile::save()

6.3.24.3.6 PIRpTIDefsConfig Class

Overview:

Instances of PIRpTIDefsConfig load and save basic timeline configuration files.

Export Control: Public

Inheritance Relationships:

Inherits from `PIRpTIConfig`

Attributes:

myHandleColor: HString

handle colorname

myHeight: int

overall, region, and subregion dimensions

myHorScrollInc: double

horizontal scroll increment

myHorZoomInc: double

horizontal zoom increment

myIntvlInit: HString

display interval initial string

myIntvlOpts: HObjList

list of interval options (stored as a list of strings) for display

myLabelColor: HString

label colorname

myLblFontFamily: HString

label font parameters

myLblFontSize: int

label font parameters

myLblFontStyle: HString

label font parameters

myLblRegColor: HString

label region colorname

myLblRegWidth: int

overall, region, and subregion dimensions

myMainRegColor: HString

main region colorname

myOvrsFactor: double

oversubscription display factor

myScFontFamily: HString

scalable font parameters

myScFontInc: int

scalable font parameters

myScFontMaxSize: int

scalable font parameters

myScFontMinSize: int

scalable font parameters

myScFontStyle: HString

scalable font parameters

myScrollRegColor: HString

scroll region colorname

myScrollRegWidth: int

overall, region, and subregion dimensions

mySubRegsHeight: int

overall, region, and subregion dimensions

myTextColor: HString

text colorname

myTsRegColor: HString

timestamp region colorname

myTsRegHeight: int

overall, region, and subregion dimensions

myWidth: int

overall, region, and subregion dimensions

myXtl: Xtl*

my owning xtl

myZbRegColor: HString

zoombar region colorname

myZbRegHeight: int

overall, region, and subregion dimensions

Constructors and Destructor:

```
PlRpTlDefsConfig(Xtl* );
```

constructor. calls base class constructor with default arguments. argument is owning xtl

```
PlRpTlDefsConfig(const PlRpTlDefsConfig& orig);
```

copy - disabled

```
~PlRpTlDefsConfig();
```

destructor

Operations:

- configure

```
virtual int configure(int argc, char** argv);
```

yet another signature for configure

- configure

```
virtual int configure();
```

configure. here because I overrode the other configure above.

- configure

```
virtual int configure(const char* configInfo);
```

configure. override to empty any existing collection of interval options in preparation to read in new ones. then do base class behavior.

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlDefsConfig );
```

Instances of PlRpTlDefsConfig load and save basic timeline configuration files.

- getConfig

```
virtual int getConfig();
```

get any config info that I am storing which might have changed. overrides PlRpTlConfig::getConfig()

- handleColor

```
virtual const HString& handleColor() const;
```

get my handle colorname

- handleKeyword

```
virtual int handleKeyword(const HString& kywd);
```

handle keyword. look in the file for the specific keywords associated with my variables and set my variables to the following values. overrides ConfigurationFile::handleKeyword()

- height

```
virtual int height() const;
```

get my height dimension

- horScrollInc

```
virtual double horScrollInc() const;
```

get my horizontal scroll increment

- horZoomInc

```
virtual double horZoomInc() const;
```

get my horizontal zoom increment

- intvlInit

```
virtual const HString& intvlInit() const;
```

get my display interval initial string

- intvlOpts

```
virtual HObjCollection& intvlOpts();
```

get my collection of display interval options

- labelColor

```
virtual const HString& labelColor() const;
```

get my label colorname

- lblFontFamily

```
virtual const HString& lblFontFamily() const;
```

get my label font family

- lblFontSize

```
virtual int lblFontSize() const;
```

get my label font size

- lblFontStyle

```
virtual const HString& lblFontStyle() const;
```

get my label font style

- lblRegColor

```
virtual const HString& lblRegColor() const;
```

get my label region colorname

- lblRegWidth

```
virtual int lblRegWidth() const;
```

get my label region dimension

- mainRegColor

```
virtual const HString& mainRegColor() const;
```

get my main region colorname

- operator =

```
PlRpTlDefsConfig& operator =(const PlRpTlDefsConfig& orig);
```

assignment - disabled

- ovrFactor

```
virtual double ovrFactor() const;
```

get my oversubscription display factor. this is the factor by which ptl multiplies the actual maximum values for power and data resources to tell each histogram or graph event what their maximum vertical values should be.

- saveConfig

```
virtual int saveConfig(ostream& str);
```

save my current config info to the given stream.

- scFontFamily

```
virtual const HString& scFontFamily() const;
```

get my scalable font family

- scFontInc

```
virtual int scFontInc() const;
```

get my scalable font increment

- scFontMaxSize

```
virtual int scFontMaxSize() const;
```

get my scalable font maximum size

- scFontMinSize

```
virtual int scFontMinSize() const;
```

get my scalable font minimum size

- scFontStyle

```
virtual const HString& scFontStyle() const;
```

get my scalable font family

- scrollRegColor

```
virtual const HString& scrollRegColor() const;
```

get my scroll region colorname

- scrollRegWidth

```
virtual int scrollRegWidth() const;
```

get my scroll region dimension

- subRegsHeight

```
virtual int subRegsHeight() const;
```

get my subregion region dimension

- textColor

```
virtual const HString& textColor() const;
```

get my text colorname

- tsRegColor

```
virtual const HString& tsRegColor() const;
```

get my timestamp region colorname

- tsRegHeight

```
virtual int tsRegHeight() const;
```

get my timestamp region dimension

- width

```
virtual int width() const;
```

get my width dimension

- zbRegColor

```
virtual const HString& zbRegColor() const;
```

get my zoombar region colorname

- zbRegHeight

```
virtual int zbRegHeight() const;
get my zoombar region dimension
```

6.3.24.3.7 PIRpTlExec Class

Overview:

This is a "Singleton" that allows any GUI to bring-up one or more Reservation Time-Line GUIs...

Export Control: Public

Inheritance Relationships:

Attributes:

```
_childpid: pid_t
```

This is a "Singleton" that allows any GUI to bring-up one or more Reservation Time-Line GUIs...

```
_maxprocs: unsigned int
```

general pupose use

Constructors and Destructor:

```
PIRpTlExec(unsigned int maxprocs);
```

The ctor single parameter indicates the maximum number of allowed T-L GUIs.

```
virtual ~PIRpTlExec();
```

The dtor should kill all the T-L GUIs

Operations:

- execTL

```
int execTL(const HDateTime& start, const HDateTime& end, const
char* configFile, const char* plan);
```

Performs the fork and execl

- killAll

```
void killAll();
```

kill all T-L GUIs (children)

- validExec

```
int validExec(RWCString& path);
```

Checks for existence of the executable binary

- validTLConfigFile

```
int validTLConfigFile(const RWCString file);
```

If no T-L config file exists, this creates one

6.3.24.3.8 PIRpTIFileDiag Class

Overview:

Instances of PIRpTIFileDiag are base classes dialogs for handling the loading and saving of configuration files.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpTIFileSelDiag

Attributes:

```
myAppl: PIRpTlAppl*
```

The ptl application

```
myCreatorGone: int
```

Tells me if my creator is still around.

```
myCreator: PIRpTlPanel*
```

My creator

Constructors and Destructor:

```
PIRpTlFileDiag(PIRpTlPanel* creator, PIRpTlAppl* appl);
```

Constructor. arguments are my creator and the PIRpTlAppl

```
PIRpTlFileDiag(const PIRpTlFileDiag& orig);
```

Copy - disabled

```
~PIRpTlFileDiag();
```

Destructor

Operations:

- cancel

```
virtual void cancel(XtPointer callData);
```

Cancel button pushed. Hide myself. Overrides DSelectionBox::cancel()

- creatorGone

```
virtual void creatorGone(int tf);
```

Set whether or not my creator is gone. If my creator gets destroyed first, it will tell me

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlFileDiag );
```

Instances of PIRpTlFileDiag are base classes dialogs for handling the loading and saving of configuration files.

- operator =

```
PlRpTlFileDiag& operator =(const PlRpTlFileDiag& orig);
```

Assignment - disabled

- show

```
virtual void show();
```

Show. Overridden to set the pattern and the initial selection overrides PIRpTlFileSelDiag::show()

6.3.24.3.9 PIRpTlFileSelDiag Class

Overview:

This is a derived DFileSelectBox that AUTOMATICALLY creates a FileSelectionDialog (i.e. both the FileSelectionBox widget and its DialogShell parent are created no matter what create function you call.) In addition, this class adds behavior to make sure the DialogShell parent behaves just like a DPopup.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PlRpTlFileSelDiag();
```

Constructor.

```
PlRpTlFileSelDiag(const PlRpTlFileSelDiag& orig);
```

Copy - disabled

```
~PlRpTlFileSelDiag();
```

Destructor

Operations:

- catchCloseEvent

```
virtual void catchCloseEvent();
```

Set up to intercept the window manager close event on the DialogShell, to hide me.

- createFileSelectionBox

```
virtual int createFileSelectionBox(const char* name, DWidget* parent, Arg* arglist, Cardinal argcount, int manage);
```

Create File Selection Box routine.

- create

```
virtual int create(const char* name, DWidget* parent);
```

Create routines. Overridden to make sure a FileSelectionDialog is actually created, not just the FileSelectionBox widget.

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlFileSelDiag );
```

This is a derived DFileSelectBox that AUTOMATICALLY creates a FileSelectionDialog (i.e. both the FileSelectionBox widget and its DialogShell parent are created no matter what create function you call.) In addition, this class adds behavior to make sure the DialogShell parent behaves just like a DPopup.

- hide

```
virtual void hide();
```

Hide me (i.e. my DialogShell) from the desktop

- operator =

```
PlRpTlFileSelDiag& operator =(const PlRpTlFileSelDiag& orig);
```

Assignment - disabled

- show

```
virtual void show();
```

Show me (i.e. my DialogShell) on the desktop

- winManagerClose

```
static void winManagerClose(Widget protocolWidget, XtPointer
clientData, XtPointer callData);
```

Static callback for window manager close events.

6.3.24.3.10 PIRpTILoadDiag Class

Overview:

Instances of PIRpTILoadDiag are dialogs for loading saved timeline configuration files.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpTIFileDiag

Attributes:

Constructors and Destructor:

```
PlRpTILoadDiag(PlRpTIPanel* creator, PlRpTlAppl* appl);
```

Constructor. arguments are my creator and the PIRpTlAppl

```
PlRpTILoadDiag(const PlRpTILoadDiag& orig);
```

Copy - disabled

```
~PlRpTILoadDiag();
```

Destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTILoadDiag );
```

Instances of PIRpTILoadDiag are dialogs for loading saved timeline configuration files.

- ok

```
virtual void ok(XtPointer callData);
```

ok button pushed. Tell the ptl appl to load and switch to a new configuration Overrides DSelectionBox::ok()

- operator =

```
PlRpTlLoadDiag& operator =(const PlRpTlLoadDiag& orig);
```

Assignment - disabled

6.3.24.3.11 PIRpTlMainRegion Class

Overview:

A PIRpTlMainRegion is a `_TLMainRegion_` that implements "timeline" mouse tracking behavior. As part of this, it adds the functionality of displaying the current activity region that the mouse is on.

Export Control: Public

Inheritance Relationships:

Attributes:

```
mouseX: unsigned long
```

my mouse X position

```
myTLEvent: TLEvent*
```

my TLEvent

```
myTs: TLTimeScaleAbs*
```

my TLTimeScaleAbs timescale

```
myXtl: Xtl*
```

my owning xtl

Constructors and Destructor:

```
PlRpTlMainRegion(const PlRpTlMainRegion& orig);
```

Restricted access for copying and assignment. copy constructor

```
PlRpTlMainRegion(Xtl* , int x, int y, int w, int h);
```

Constructor for this class - arguments are parent Xtl and dimensions.

```
PlRpTlMainRegion(Xtl* );
```

Constructor - argument is parent Xtl.

```
~PlRpTlMainRegion();
```

destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlMainRegion );
```

A PlRpTlMainRegion is a _TlMainRegion_ that implements "timeline" mouse tracking behavior. As part of this, it adds the functionality of displaying the current activity region that the mouse is on.

- handleMouseEvent

```
virtual void handleMouseEvent(Event* );
```

mouse track handling

- handleSimpleEvent

```
virtual const HString handleSimpleEvent() const;
```

returns TlSimpleEvent info to be displayed

- operator =

```
PlRpTlMainRegion& operator =(const PlRpTlMainRegion& orig);
```

assignment operator

- setHandlerEvents

```
virtual void setHandlerEvents();
```

Set up events which I will catch overrides TlMainRegion::setHandlerEvents().

6.3.24.3.12 PIRpTIPanel Class

Overview:

This class, which is derived from TlPanel, adds capabilities for interactively changing colors and resources, and for saving and loading from different configuration files.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myAppl: PlRpTlAppl*
```

The ptl application

```
myColorChanger: PlRpTlColChngr*
```

The color changer

`myDisplayMenu: DPullDownMenu*`

The display menu

`myFileMenu: DPullDownMenu*`

The file menu

`myLoadDiag: PlRpTlLoadDiag*`

The load config dialog

`myPlanButtonList: HObjList`

The list of plan menu buttons

`myPlansMenu: DPullDownMenu*`

The plans menu

`myPlanWinChgr: PlRpTlPlanWinChgr*`

The plan window changer

`myRsEditor: PlRpTlRsEditor*`

The resource editor

`mySaveDiag: PlRpTlSaveDiag*`

The save config dialog

`myTimeMenu: DPullDownMenu*`

The time menu

Constructors and Destructor:

```
PlRpTlPanel(PlRpTlAppl* appl, PlRpTlXtl* xtl);
```

Constructor. Arguments are the ptl application and the xtl.

```
PlRpTlPanel(const PlRpTlPanel& orig);
```

Copy - disabled

```
~PlRpTlPanel();
```

Destructor.

Operations:

- addToPlansMenu

```
virtual int addToPlansMenu();
```

Add a button to my Plans menu for the given plan name

- clearPlansMenu

```
virtual int clearPlansMenu();
```

Clear all buttons out of my Plans menu

- colorChangerGone

```
virtual void colorChangerGone();
```

Tell me that the color changer is gone

- colorChanger

```
virtual void colorChanger(PlRpTlPanelCbPushBtn* );
```

Create color changer popup. Callback for display menu button item.

- createColorChanger

```
virtual PlRpTlColChngr* createColorChanger();
```

Create the color changer

- createDisplayMenuItems

```
virtual int createDisplayMenuItems();
```

Create the push buttons on the display menu.

- createDisplayMenu

```
virtual int createDisplayMenu();
```

Create my display button.

- createFileMenuItems

```
virtual int createFileMenuItems();
```

Create the push buttons on the configuration menu.

- createFileMenu

```
virtual int createFileMenu();
```

Create my configuration menu.

- createLoadDiag

```
virtual PlRpTlLoadDiag* createLoadDiag();
```

Create the load config dialog

- createMenus

```
virtual int createMenus();
```

Create panel menus. Behavior is to call base class function and then add the config and display buttons. overrides TlPanel::createMenus()

- createPlansMenuItems

```
virtual int createPlansMenuItems();
```

Create the initial push buttons on the plans menu.

- createPlansMenu

```
virtual int createPlansMenu();
```

Create my plans menu

- createPlanWinChgr

```
virtual PlRpTlPlanWinChgr* createPlanWinChgr();
```

Create the plan window changer

- createRsEditor

```
virtual PlRpTlRsEditor* createRsEditor();
```

Create the resource editor

- createSaveDiag

```
virtual PlRpTlSaveDiag* createSaveDiag();
```

Create the save config dialog

- createTimeMenuItems

```
virtual int createTimeMenuItems();
```

Create the push buttons on the time menu

- createTimeMenu

```
virtual int createTimeMenu();
```

Create my time menu

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlPanel );
```

This class, which is derived from TlPanel, adds capabilities for interactively changing colors and resources, and for saving and loading from different configuration files.

- loadConfig

```
virtual void loadConfig(PlRpTlPanelCbPushBtn* );
```

Load a configuration. Callback for file menu button item.

- loadDiagGone

```
virtual void loadDiagGone();
```

Tell me that the load config dialog is gone

- operator =

```
PlRpTlPanel& operator =(const PlRpTlPanel& orig);
```

Assignment - disabled

- planSnapper

```
virtual void planSnapper(PlRpTlPanelCbPushBtn* );
```

Handle requested plan snap. Callback for plans menu button items.

- planWinChanger

```
virtual void planWinChanger(PlRpTlPanelCbPushBtn* );
```

Create plan window changer popup. Callback for time menu button item.

- planWinChgrGone

```
virtual void planWinChgrGone();
```

Tell me that my plan window changer is gone

- printTimeline

```
virtual void printTimeline(PlRpTlPanelCbPushBtn* );
```

Print the timeline by spawning a postscript timeline. Callback for file menu button item.

- quit

```
virtual void quit(PlRpTlPanelCbPushBtn* );
```

Callback for file menu quit button item.

- reconfigure

```
virtual int reconfigure();
```

Reconfigure. Tell any of my displays that need to know about coming up on a new configuration.

- remFromPlansMenu

```
virtual int remFromPlansMenu();
```

Remove the named button from the Plans menu returns TRUE if the button was found AND removed

- rsEditorGone

```
virtual void rsEditorGone();
```

Tell me that the resource editor is gone

- rsrcChanger

```
virtual void rsrcChanger(PlRpTlPanelCbPushBtn* );
```

Create resource changer popup. Callback for display menu button item.

- saveConfig

```
virtual void saveConfig(PlRpTlPanelCbPushBtn* );
```

Save the current configuration. Callback for file menu button item.

- saveDiagGone

```
virtual void saveDiagGone();
```

Tell me that the save config dialog is gone

6.3.24.3.13 PIRpTIPlanWinChgr Class

Overview:

Instances of PIRpTIPlanWinChgr are dialogs for selecting a plan.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myAppl: PlRpTlAppl*
```

the PIRpTIXtl appl

```
myCreatorGone: int
```

tells me if my creator is still around

```
myCreator: PlRpTlPanel*
```

my creator panel

```
myPresForm: DForm*
```

my presentation area form

```
myPt1: PlRpTlXtl*
```

the PIRpTIXtl

`myStartTime: DDateTime*`

my start time item

`myStopTime: DDateTime*`

my stop time item

Constructors and Destructor:

```
PlRpTlPlanWinChgr(PlRpTlPanel* creator, PlRpTlAppl* appl,  
PlRpTlXtl* PlRpTlXtl);
```

constructor. arguments are my creator, the PIRpTlAppl, and PIRpTlXtl

```
PlRpTlPlanWinChgr(const PlRpTlPlanWinChgr& orig);
```

copy - disabled

```
~PlRpTlPlanWinChgr();
```

destructor

Operations:

- applyChanges

```
virtual void applyChanges();
```

apply changes. tell PIRpTlXtl to switch to the new total interval overrides
DApplyDialog::applyChanges()

- createDispObjs

```
virtual int createDispObjs();
```

create the display objects that make up my presentation area.

- createPresentationArea

```
virtual int createPresentationArea();
```

make my presentation area. overrides DApplyDialog::createPresentationArea()

- createStartTime

```
virtual int createStartTime();
```

create the start time item

- createStopTime

```
virtual int createStopTime();
```

create the stop time item

- creatorGone

```
virtual void creatorGone(int tf);
```

set whether or not my creator is gone. if my creator gets destroyed first, it will tell me

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlPlanWinChgr );
```

Instances of PlRpTlPlanWinChgr are dialogs for selecting a plan.

- manageAllChildren

```
virtual void manageAllChildren();
```

ask all of pieces to manageChild. does the pieces I created plus my base class. overrides DApplyDialog::manageAllChildren()

- operator =

```
PlRpTlPlanWinChgr& operator =(const PlRpTlPlanWinChgr& orig);
```

assignment - disabled

- reconfigure

```
virtual int reconfigure();
```

reconfigure. reinitialize the available and viewed lists

- setStartStopTimes

```
virtual int setStartStopTimes();
```

initialize the start and stop times with the current total interval

- setStartTimePos

```
virtual void setStartTimePos();
```

set the position of the start time item

- setStopTimePos

```
virtual void setStopTimePos();
```

set the position of the stop time item

- show

```
virtual void show();
```

show me on the desktop. override to set start and stop times to current total interval. then do base class behavior

6.3.24.3.14 PIRpTlReptExec Class

Overview:

This is a "Singleton" that allows any GUI to bring-up one or more Reservation Report GUIs...

Export Control: Public

Inheritance Relationships:

Attributes:

`_childpid: pid_t`

This is a "Singleton" that allows any GUI to bring-up one or more Reservation Report GUIs...

`_maxprocs: unsigned int`

general pupose use

Constructors and Destructor:

`PlRpTlReptExec(unsigned int maxprocs);`

The ctor single parameter indicates the maximum number of allowed Report GUIs.

`virtual ~PlRpTlReptExec();`

The dtor should kill all the Reservation Report GUIs

Operations:

- execRept

`int execRept();`

Performs the fork and execl

- killAll

`void killAll();`

kill all report GUIs

- validExec

`int validExec(RWCString& path);`

Checks for existence of the executable binary

6.3.24.3.15 PIRpTIRsConfig Class

Overview:

Instances of PIRpTIRsConfig are responsible for loading a set of names of resources to be displayed on the timeline.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpTIConfig

Attributes:

`myAppl: PIRpTIAppl*`
my owning PIRpTIAppl

Constructors and Destructor:

```
PIRpTIRsConfig(PIRpTIAppl* );
```

constructor. calls base class constructor with default arguments. argument is owning PIRpTIAppl

```
PIRpTIRsConfig(const PIRpTIRsConfig& orig);
```

copy - disabled

```
~PIRpTIRsConfig();
```

destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpTIRsConfig );
```

Instances of PIRpTIRsConfig are responsible for loading a set of names of resources to be displayed on the timeline.

- handleKeyword

```
virtual int handleKeyword(const HString& kywd);
```

handle keyword. for each resource or resource type in the config file, ask the srm client noid to get the appropriate resource(s) from the resource allocator. overrides ConfigurationFile::handleKeyword()

- handleRsType

```
virtual int handleRsType();
```

handle a resource type in the config file

- handleRs

```
virtual int handleRs();
```

handle an individual resource in the config file

- operator =

```
PIRpTlRsConfig& operator =(const PIRpTlRsConfig& orig);
```

assignment - disabled

- saveConfig

```
virtual int saveConfig(ostream& str);
```

save the current set of resources to the given stream overrides PIRpTIConfig::saveConfig()

6.3.24.3.16 PIRpTIRsEditor Class

Overview:

Instances of PIRpTIRsEditor are dialogs for choosing the set of resources to be displayed on the timeline.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myAddBtn: PIRpTlRsEditorCbPushBtn*
```

add resource to viewed list

```
myAppl: PIRpTlAppl*
```

the ptl appl

```
myAvailLbl: DLabel*
```

my available list labels

```
myAvailList: PIRpTlRsEditorCbList*
```

my scrolling list of available resources

```
myCreatorGone: int
```

tells me if my creator is still around

myCreator: PIRpTlPanel*

my creator panel

myDelBtn: PIRpTlRsEditorCbPushBtn*

delete resource from viewed list

myDownBtn: PIRpTlRsEditorCbArrowBtn*

re-order resource list (move selected down)

myPresForm: DForm*

my presentation area form

myUpBtn: PIRpTlRsEditorCbArrowBtn*

re-order resource list (move selected up)

myViewedLbl: DLabel*

my viewed list labels

myViewedList: PIRpTlRsEditorCbList*

my scrolling list of viewed resources

Constructors and Destructor:

```
PIRpTlRsEditor(PIRpTlPanel* creator, PIRpTlAppl* appl);
```

constructor. arguments are my creator and the PIRpTlAppl

```
PIRpTlRsEditor(const PIRpTlRsEditor& orig);
```

copy - disabled

```
~PIRpTlRsEditor();
```

destructor

Operations:

- addCb

```
virtual void addCb(PIRpTlRsEditorCbPushBtn* );
```

callback for add button

- applyChanges

```
virtual void applyChanges();
```

apply changes. tell the ptl appl to switch resources. overrides DApplyDialog::applyChanges()

- availListSelCb

```
virtual void availListSelCb(PlRpTlRsEditorCbList* );
```

select callback for available resource list

- createAddBtn

```
virtual int createAddBtn();
```

create the add-to-viewed resource button

- createAvailLbl

```
virtual int createAvailLbl();
```

create the label for the available resource list

- createAvailList

```
virtual int createAvailList();
```

create the available resource list

- createBtns

```
virtual int createBtns();
```

create the control buttons

- createDelBtn

```
virtual int createDelBtn();
```

create the delete-from-viewed resource button

- createDispObjs

```
virtual int createDispObjs();
```

create the display objects that make up my presentation area.

- createDownBtn

```
virtual int createDownBtn();
```

create the move-down in the viewed list button

- createPresentationArea

```
virtual int createPresentationArea();
```

make my presentation area. overrides DApplyDialog::createPresentationArea()

- createUpBtn

```
virtual int createUpBtn();
```

create the move-up in the viewed list button

- createViewedLbl

```
virtual int createViewedLbl();
```

create the label for the viewed resource list

- createViewedList

```
virtual int createViewedList();
```

create the viewed resource list

- creatorGone

```
virtual void creatorGone(int tf);
```

set whether or not my creator is gone. if my creator gets destroyed first, it will tell me

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlRsEditor );
```

Instances of PlRpTlRsEditor are dialogs for choosing the set of resources to be displayed on the timeline.

- delCb

```
virtual void delCb(PlRpTlRsEditorCbPushBtn* );
```

callback for delete button

- downCb

```
virtual void downCb(PlRpTlRsEditorCbArrowBtn* );
```

callback for down button

- initAvailList

```
virtual int initAvailList();
```

initialize the list with all the resources available

- initViewedList

```
virtual int initViewedList();
```

initialize the list with the current resources being viewed

- manageAllChildren

```
virtual void manageAllChildren();
```

ask all of pieces to manageChild. does the pieces I created plus my base class. overrides DApplyDialog::manageAllChildren()

- operator =

```
PlRpTlRsEditor& operator =(const PlRpTlRsEditor& orig);
```

assignment - disabled

- reconfigure

```
virtual int reconfigure();
```

reconfigure. reinitialize the available and viewed lists

- setAddBtnPos

```
virtual void setAddBtnPos();
```

set the add button position

- setAvailLblPos

```
virtual void setAvailLblPos();
```

set the position of available list label

- setAvailListPos

```
virtual void setAvailListPos();
```

set the position of the available list

- setDelBtnPos

```
virtual void setDelBtnPos();
```

set the delete button position

- setDownBtnPos

```
virtual void setDownBtnPos();
```

set the down button position

- setUpBtnPos

```
virtual void setUpBtnPos();
```

set the up button position

- setViewedLblPos

```
virtual void setViewedLblPos();
```

set the position of viewed list label

- setViewedListPos

```
virtual void setViewedListPos();
```

set the position of the viewed list

- show

```
virtual void show();
```

show me on the desktop. override to init viewed list each time, then do base class behavior

- upCb

```
virtual void upCb(PlRpTlRsEditorCbArrowBtn* );
```

callback for up button

- viewedListSelCb

```
virtual void viewedListSelCb(PlRpTlRsEditorCbList* );
```

select callback for viewed resource list

6.3.24.3.17 PIRpTISaveDiag Class

Overview:

Instances of PIRpTISaveDiag are dialogs for saving timeline configuration files.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpTIFileDiag

Attributes:

Constructors and Destructor:

```
PlRpTISaveDiag(PlRpTlPanel* creator, PlRpTlAppl* appl);
```

constructor. arguments are my creator and the PIRpTlAppl

```
PlRpTISaveDiag(const PlRpTISaveDiag& orig);
```

copy - disabled

```
~PlRpTISaveDiag();
```

destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTISaveDiag );
```

Instances of PIRpTISaveDiag are dialogs for saving timeline configuration files.

- ok

```
virtual void ok(XtPointer callData);
```

ok button pushed. tell the ptl appl to save configuration overrides DSelectionBox::ok()

- operator =

```
PlRpTlSaveDiag& operator =(const PlRpTlSaveDiag& orig);
```

cassignment - disabled

6.3.24.3.18 PIRpTlSimpleEvent Class

Overview:

Instances of PIRpTlSimpleEvent provide base class behavior that all timeline simple events will provide. Derived PIRpTlSimpleEvent(s) have a resource state, know how to determine that state, determine if it is a valid state and can set the type of label to display on the timeline.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myState: RRsState*
```

pointer to my associated state

Constructors and Destructor:

```
PlRpTlSimpleEvent(Xtl* );
```

constructor

```
PlRpTlSimpleEvent(const PlRpTlSimpleEvent& orig);
```

copy constructor

```
~PlRpTlSimpleEvent();
```

destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlSimpleEvent );
```

Instances of PIRpTlSimpleEvent provide base class behavior that all timeline simple events will provide. Derived PIRpTlSimpleEvent(s) have a resource state, know how to determine that state, determine if it is a valid state and can set the type of label to display on the timeline.

- isValidState

```
virtual int isValidState(RRsState* );
```

return TRUE if the given RRsState is one I will accept. pure virtual - derived classes must override

- operator =

```
PlRpTlSimpleEvent& operator =(const PlRpTlSimpleEvent& orig);
```

assignment operator

- overlap

```
virtual int overlap(const char* rsrcNm, const HEpochInterval& intvl);
```

return TRUE if I belong to the specified resource and overlap the given interval

- rsrcNm

```
virtual void rsrcNm(const char* );
```

Set my associated resource name, which corresponds to the name of my subregion

- rsrcNm

```
virtual const HString& rsrcNm() const;
```

Get my associated resource name, which corresponds to the name of my subregion

- setLabelAndType

```
virtual void setLabelAndType();
```

set my label and type according to myState pure virtual - derived classes must override

- state

```
virtual RRsState* state() const;
```

get my associated state

- state

```
virtual int state(RRsState* );
```

set my associated state and determine label and type for this event. if state is accepted by event, return TRUE; if state is not accepted by event, returns FALSE;

6.3.24.3.19 PIRpTISliderPup Class

Overview:

Instances of PIRpTISliderPup are popup windows which display a start time and stop time for the corresponding plan region.

Export Control: Public

Inheritance Relationships:

Attributes:

`myEndLabel: DLabel*`

my end time label

`myRowColumn: DRowColumn*`

my row-column manager widget

`myStartLabel: DLabel*`

my start time label

Constructors and Destructor:

`PIRpTISliderPup();`

default constructor

`PIRpTISliderPup(const PIRpTISliderPup& orig);`

copy - disabled

`~PIRpTISliderPup();`

destructor

Operations:

- createDisplay

`virtual int createDisplay();`

create my display.

- DECLARE_LOCAL_CLASS

`int DECLARE_LOCAL_CLASS(PIRpTISliderPup);`

Instances of PIRpTISliderPup are popup windows which display a start time and stop time for the corresponding plan region.

- manageAllChildren

```
virtual void manageAllChildren();
```

manage all of my pieces

- operator =

```
PIRpTISliderPup& operator =(const PIRpTISliderPup& orig);
```

assignment - disabled

- setEnd

```
virtual void setEnd(const HEpochTime& time);
```

set end time to the given epoch time

- setStart

```
virtual void setStart(const HEpochTime& time);
```

set start time to the given epoch time

6.3.24.3.20 PIRpTISrmNoid Class

Overview:

Instances of PIRpTISrmNoid are timeline resource model client noids. A PIRpTISrmNoid knows how to establish a connection with a resource model, retrieve initial information and is notified when new updates are received by the noids agent

Export Control: Public

Inheritance Relationships:

Inherits from PIRpClSrmNoid

Attributes:

```
myParent: PIRpTlAppl*
```

My parent appl

Constructors and Destructor:

```
PIRpTISrmNoid(const PIRpTISrmNoid& orig);
```

Copy constructor

```
PIRpTISrmNoid(PIRpTlAppl* appl);
```

Constructor specifying parent

```
PIRpTlSrmNoid();
```

Default constructor

```
~PIRpTlSrmNoid();
```

Destructor

Operations:

- connEstablished

```
virtual int connEstablished();
```

Notification connection established for first time overridden from PIRpClSrmNoid::connEstablished()

- connReestablished

```
virtual int connReestablished();
```

Notification reestablished after being lost

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpTlSrmNoid );
```

Instances of PIRpTlSrmNoid are timeline resource model client noids. A PIRpTlSrmNoid knows how to establish a connection with a resource model, retrieve initial information and is notified when new updates are received by the noids agent

- handlePlansGetRsp

```
virtual int handlePlansGetRsp(HMsgConn* curCon, HAddress* curAdr, PIRpClMsgGetPlansRsp* curMsg);
```

Handle getting plans into our local plan pool and then getting a separate list of plan names

- handlePlanSnap

```
virtual int handlePlanSnap(HMsgConn* curCon, HAddress* curAdr, SMsgPlanSnap* curMsg);
```

Ask my parent appl to handle a request to snap to a new plan.

- handleRsGetNmsRsp

```
virtual int handleRsGetNmsRsp(HMsgConn* curCon, HAddress* curAdr, PIRpClMsgGetRsNmsRsp* curMsg);
```

Turn over the list of resource names to my parent. overrides PIRpClSrmNoid::handleRsGetNmsRsp()

- installRsGetRs

```
virtual int installRsGetRs(RResource* rs);
```

Tell my parent about a resource that has just been loaded into my resource pool.

- installRsState

```
virtual int installRsState(RUpdState* anUpdate);
```

Tell my parent about the state update I just received. overrides PIRpClSrmNoid::installRsState()

- operator =

```
PIRpTlSrmNoid& operator =(const PIRpTlSrmNoid& orig);
```

Assignment operator

- parent

```
PIRpTlAppl* parent() const;
```

- parent

```
void parent(PIRpTlAppl* appl);
```

Set/get my Appl parent

6.3.24.3.21 PIRpTlStatusBar Class

Overview:

Instances of PIRpTlStatusBar provide a line at the bottom of the timeline for displaying messages.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myDisplayLabel: DLabel*
```

my display label

```
myXtl: Xtl*
```

my owning xtl

Constructors and Destructor:

```
PIRpTlStatusBar(Xtl* );
```

constructor. given a pointer to my owning xtl

```
PlRpTlStatusBar(const PlRpTlStatusBar& orig);
```

copy - disabled

```
~PlRpTlStatusBar();
```

destructor

Operations:

- clear

```
virtual void clear();
```

clear.

- createMyLabel

```
virtual int createMyLabel();
```

create status display label

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlStatusBar );
```

Instances of PlRpTlStatusBar provide a line at the bottom of the timeline for displaying messages.

- displayMesg

```
virtual void displayMesg(const HString& mesg);
```

display received message

- init

```
virtual int init();
```

initialize. create the form and its child, DLabel

- operator =

```
PlRpTlStatusBar& operator =(const PlRpTlStatusBar& orig);
```

assignment - disabled

- positionWidgets

```
virtual int positionWidgets();
```

position widgets

6.3.24.3.22 PIRpTlStplConfig Class

Overview:

Instances of PIRpTlStplConfig handle stipple keywords in configuration files.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpTlConfig

Attributes:

myXtl: Xtl*
my owning xtl

Constructors and Destructor:

```
PIRpTlStplConfig(Xtl* );  
constructor. calls base class constructor with default arguments. argument is owning xtl
```

```
PIRpTlStplConfig(const PIRpTlStplConfig& orig);  
copy - disabled
```

```
~PIRpTlStplConfig();  
destructor
```

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpTlStplConfig );  
Instances of PIRpTlStplConfig handle stipple keywords in configuration files.
```

- handleKeyword

```
virtual int handleKeyword(const HString& kywd);  
handle keyword. process "stipple" keyword
```

- handleType

```
virtual int handleType();  
load TLTypes with type name and TLStplMapper with the type name and associated bitmap file.
```

- operator =

```
PIRpTlStplConfig& operator =(const PIRpTlStplConfig& orig);
```

assignment - disabled

- saveConfig

```
virtual int saveConfig(ostream& str);
```

save my current config info to the given stream

6.3.24.3.23 PIRpTlStplEvent Class

Overview:

Instances of PIRpTlStplEvent are stipple events associated with resource states.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpTlStplEv

Attributes:

```
myState: RRsState*
```

pointer to my associated state

Constructors and Destructor:

```
PIRpTlStplEvent(Xtl* xtl);
```

default constructor

```
PIRpTlStplEvent(const PIRpTlStplEvent& orig);
```

copy constructor

```
~PIRpTlStplEvent();
```

destructor

Operations:

- createView

```
virtual TLEventView* createView(TLEventSubRegion* );
```

create an LtStplView overrides PIRpTlStplEv::createView

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpTlStplEvent );
```

Instances of PIRpTlStplEvent are stipple events associated with resource states.

- isValidState

```
virtual int isValidState(RRsState* );
```

return TRUE if the given RRsState is one I will accept.

- operator =

```
PIRpTlStplEvent& operator =(const PIRpTlStplEvent& orig);
```

assignment operator

- overlap

```
virtual int overlap(const char* rsrcNm, const HEpochInterval& intvl);
```

return TRUE if I belong to the specified resource and overlap the given interval

- rsrcNm

```
virtual void rsrcNm(const char* );
```

Set my associated resource name, which corresponds to the name of my subregion

- rsrcNm

```
virtual const HString& rsrcNm() const;
```

Get my associated resource name, which corresponds to the name of my subregion

- setLabelAndType

```
virtual void setLabelAndType();
```

set my label and type according to myState

- state

```
virtual RRsState* state() const;
```

get my associated state

- state

```
virtual int state(RRsState* );
```

set my associated state and determine label and type for this event. if state is accepted by event, return TRUE; if state is not accepted by event, returns FALSE;

6.3.24.3.24 PIRpTlStplEv Class

Overview:

Instances of PIRpTlStplEv are stipple events.

Export Control: Public

Inheritance Relationships:

Attributes:

`myStipple: TLStipple*`

my TLStipple containing the bitmap I am to draw

`ownStippleFlag: int`

flag indicating whether I own the stipple or not. default is that I DO NOT own the stipple.

Constructors and Destructor:

`PlRpTlStplEv(Xtl* xtl);`

constructor. argument is "my owning" xtl

`PlRpTlStplEv(const PlRpTlStplEv& orig);`

copy - disabled

`~PlRpTlStplEv();`

destructor

Operations:

- createView

`virtual TLEventView* createView(TLEventSubRegion*);`

create a PIRpTlStplVw. overrides TLEvent::createView

- DECLARE_LOCAL_CLASS

`int DECLARE_LOCAL_CLASS(PlRpTlStplEv);`

Instances of PIRpTlStplEv are stipple events.

- interval

`virtual const HEpochInterval& interval() const;`

set interval set overridden to ensure that myIntvl is degenerate, i.e. of 0 duration, with time of intvl.start() overrides TSimpleEvent::interval()

- interval

`virtual void interval(const HEpochInterval& intvl);`

get interval

- operator =

```
PlRpTlStplEv& operator =(const PlRpTlStplEv& orig);
```

assignment - disabled

- ownStipple

```
virtual void ownStipple(int tf);
```

set whether or not I own the stipple. default is that I do NOT own the stipple.

- ownStipple

```
virtual int ownStipple() const;
```

get whether or not I own the stipple.

- stipple

```
virtual TLStipple* stipple();
```

get my stipple.

- stipple

```
virtual void stipple(TLStipple* tlSt);
```

set my stipple. if ownStippleFlag is TRUE, I own stipple and will delete it. if ownStippleFlag is FALSE, I do not own stipple and will not delete it.

- subRegionNm

```
virtual void subRegionNm(const char* );
```

set my associated subregion name

- subRegionNm

```
virtual const HString& subRegionNm() const;
```

get my associated subregion name

- times

```
virtual int times(HEpochTime& st, HEpochTime& en);
```

start and end time (which are the same). overridden to allow for a degenerate interval overrides TlSimpleEvent::times()

- time

```
virtual void time(const HEpochTime& tm);
```

set time this sets myIntvl to (time,time)

6.3.24.3.25 PIRpTlStplView Class

Overview:

Instances of PIRpTlStplView are views associated with stipple events.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpTlStplVw

Attributes:

Constructors and Destructor:

```
PIRpTlStplView(PIRpTlStplEv* ev);
```

constructor. argument is my owning PIRpTlStplEv

```
PIRpTlStplView(const PIRpTlStplView& orig);
```

copy - disabled

```
~PIRpTlStplView();
```

destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpTlStplView );
```

Instances of PIRpTlStplView are views associated with stipple events.

- move

```
virtual int move();
```

move. compute position on new subregion, then tell my event to move me. overrides TLEventView::move()

- operator =

```
PIRpTlStplView& operator =(const PIRpTlStplView& orig);
```

assignment - disabled

6.3.24.3.26 PIRpTlStplVw Class

Overview:

Instances of PIRpTlStplVw are views of stipple events.

Export Control: Public

Inheritance Relationships:

Attributes:

myIconRect: DXRectangle

my current icon rectangle (this rectangle moves during a drag)

myIconXHot: int

the x location of the hot spot within my icon rect, relative to the upper left corner of the icon rect the icon rect is placed so that the "hot spot" is at the exact time of the event.

myIconYHot: int

the y location of the hot spot within my icon rect, relative to the upper left corner of the icon rect

myOldIconRect: DXRectangle

my old icon rectangle, describing my last displayed position. used to replace by current icon rectangle with my last if a drag fails.

myStipple: TLStipple*

my stipple (visibility only)

myTextRect: DXRectangle

my text rectangle, for displaying the associated text. this rectangle does not move during drags.

myText: DXHString

my associated text

Constructors and Destructor:

```
PIRpTlStplVw(PIRpTlStplEv* ev);
```

constructor. argument is my owning PIRpTlStplEv

```
PIRpTlStplVw(const PIRpTlStplVw& orig);
```

copy - disabled

```
~PIRpTlStplVw();
```

destructor

Operations:

- adjPos

```
virtual void adjPos(long dx, long dy);
```

adjust position of icon rect to take into account movement as specified. overrides TLEvent::adjPos()

- containedBy

```
virtual unsigned int containedBy(HRectangle* ) const;
```

icontains behavior. overridden to take into account both the icon rect and the text rect. overrides TLEvent::contains, etc.

- contains

```
virtual unsigned int contains(HRectangle* ) const;
```

contains behavior. overridden to take into account both the icon rect and the text rect. overrides TLEvent::contains etc.

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpTlStplVw );
```

Instances of PIRpTlStplVw are views of stipple events.

- eraseOld

```
virtual void eraseOld();
```

erase old view, which is just the old icon rect plus the text. overrides TLEvent::eraseOld()

- erase

```
virtual void erase();
```

erase whole current view, which is current icon rect plus the text overrides TLEvent::erase()

- findEventRegion

```
virtual TLEventSubRegion* findEventRegion();
```

find subregion I am on by looking at position of current icon rect. overrides TLEvent::findEventRegion()

- getPos

```
virtual void getPos();
```

restore current icon rect to old icon rect. overrides TLEvent::getPos()

- handleSelectOnSelectedEvent

```
virtual void handleSelectOnSelectedEvent();
```

handle another select on an already selected event

- handleSelectOnUnselectedEvent

```
virtual void handleSelectOnUnselectedEvent();
```

handle select on a currently unselected event

- height

```
virtual void height(int );
```

set my height. overrides TLEvent::height()

- intersects

```
virtual unsigned int intersects(unsigned long x, unsigned long y) const;
```

intersects behavior. overridden to take into account both the icon rect and the text rect. overrides TLEvent::intersects

- intersects

```
virtual unsigned int intersects(const HRectangle* ) const;
```

intersects behavior. overridden to take into account both the icon rect and the text rect. overrides TLEvent::intersects etc.

- interval

```
virtual void interval(const HEpochInterval& intvl);
```

set my interval and my vis interval. overridden to ensure interval is degenerate (of 0 duration). set my interval and my vis interval. overrides TLEvent::interval and TLEvent::visInterval set my time interval

- interval

```
virtual const HEpochInterval& interval();
```

get my time interval

- moveInterval

```
virtual HEpochInterval moveInterval(TLEventSubRegion* tsr);
```

returns the time interval (which is degenerate in this case) that corresponds to where my icon rect hot spot currently is

- operator =

```
PlRpTlStplVw& operator =(const PlRpTlStplVw& orig);
```

assignment - disabled

- recalcIcon

```
virtual void recalcIcon(TLTimeScaleAbs* );
```

recalculate the position of the icon rect and the location of the hot spot relative to the icon rect.

- recalcName

```
virtual void recalcName();
```

recalculate the position of the text rect.

- recalc

```
virtual void recalc(TLTimeScaleAbs* );
```

recalculate all of my display elements. overrides TLEvent::recalc()

- repaintIcon

```
virtual int repaintIcon();
```

repaint my icon

- repaintName

```
virtual int repaintName();
```

repaint my text string overrides DspObj::repaintName()

- repaintOutline

```
virtual int repaintOutline();
```

repaint outline of current icon rectangle, (usually while dragging) overrides DspObj::repaintOutline();

- repaintSelected

```
virtual int repaintSelected();
```

repaint selected rectangle around icon overrides DspObj::repaintSelected

- repaint

```
virtual int repaint();
```

repaint myself. overrides DspObj::repaint()

- savePos

```
virtual void savePos();
```

save current icon rect in old icon rect. overrides TLEvent::savePos()

- selectView

```
virtual void selectView();
```

tell me to select. I set my selected flag and repaint, so I pick up selected rectangle around icon. overrides TLEvent::selectView()

- select

```
virtual void select();
```

handle select according to whether or not event is already selected. overrides TLEvent::select()

- stipple

```
virtual void stipple(TLStipple* );
```

set my stipple. I do NOT own it.

- stipple

```
virtual TLStipple* stipple() const;
```

get my stipple. I do NOT own it.

- unselectView

```
virtual void unselectView();
```

tell me to unselect. I reset my selected flag and repaint, so I get rid of my selected rectangle around icon. overrides TLEvent::unselectView()

- visInterval

```
virtual const HEpochInterval& visInterval();
```

get my visible time interval

6.3.24.3.27 PIRpTITypesConfig Class

Overview:

Instances of PIRpTITypesConfig handle configuration files which can contain class names which are added to the list of the timeline's known types.

Export Control: Public

Inheritance Relationships:

Inherits from PIRpTIConfig

Attributes:

```
myXtl: Xtl*
```

my owning xtl

Constructors and Destructor:

```
PlRpTlTypesConfig(Xtl* );
```

constructor. calls base class constructor with default arguments. argument is owning xtl

```
PlRpTlTypesConfig(const PlRpTlTypesConfig& orig);
```

copy - disabled

```
~PlRpTlTypesConfig();
```

destructor

Operations:

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlTypesConfig );
```

Instances of PlRpTlTypesConfig handle configuration files which can contain class names which are added to the list of the timeline's known types.

- handleKeyword

```
virtual int handleKeyword(const HString& kywd);
```

handle keyword. process "type" keyword

- handleType

```
virtual int handleType();
```

load TLTypes with type name and TLColorMapper with the type name and associated color.

- operator =

```
PlRpTlTypesConfig& operator =(const PlRpTlTypesConfig& orig);
```

assignment - disabled

- saveConfig

```
virtual int saveConfig(ostream& str);
```

save my current config info to the given stream

6.3.24.3.28 PIRpTIVaIEvent Class

Overview:

This class is a derived TLEvent which is used for displaying values versus time data graphically on a TLEventSubRegion over a specific interval of time. For example, the data may be displayed

in histogram or graph form. There is only one instance of this class per subregion for the entire interval over which data is to be displayed.

The primary behavior this class adds is the val() member function, which puts a list of values to be displayed over a specified interval into a collection provided by the user. The exact nature of the objects put in the list depends on derived classes.

Export Control: Public

Inheritance Relationships:

Attributes:

myIntvl: HEpochInterval

The entire interval over which data is to be displayed

myMaxVal: double

The maximum values to be displayed

myMinVal: double

The minimum values to be displayed

mySubRegionName: HString

The subregion that I should be displayed on

myThldVal: double

The threshold values to be displayed

myType: HString

My type, used for filtering, color, etc.

Constructors and Destructor:

```
PlRpTlValEvent(Xtl* );
```

Constructor. argument is parent xtl

```
PlRpTlValEvent(const PlRpTlValEvent& );
```

copy constructor disabled - nothing is copied

```
~PlRpTlValEvent();
```

Destructor

Operations:

- associate

```
virtual void associate(TLEventView* , TLEventSubRegion* );
```

Called to associate the subregion with the view Overrides TLEvent::associate()

- createView

```
virtual TLEventView* createView(TLEventSubRegion* );
```

Called when a new view is needed - behavior at this level is to return NULL, overriding TLEvent behavior to create TLBoxView. Each derived kind of PIRpTIValevent needs to create a particular kind of view.

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpTIValevent );
```

This class is a derived TLEvent which is used for displaying values versus time data graphically on a TLEventSubRegion over a specific interval of time. For example, the data may be displayed in histogram or graph form. There is only one instance of this class per subregion for the entire interval over which data is to be displayed.

The primary behavior this class adds is the val() member function, which puts a list of values to be displayed over a specified interval into a collection provided by the user. The exact nature of the objects put in the list depends on derived classes.

- draggable

```
virtual void draggable(unsigned int );
```

Get draggable status. Overridden to prohibit setting to TRUE. Base class behavior is for PIRpTIValevents not to be draggable.

- draggable

```
virtual unsigned int draggable() const;
```

Set draggable status. Overridden to prohibit setting to TRUE. Base class behavior is for PIRpTIValevents not to be draggable.

- interval

```
virtual void interval(const HEpochInterval& );
```

Get total interval over which I will display data

- interval

```
virtual const HEpochInterval& interval() const;
```

Set total interval over which I will display data

- isAppropriate

```
virtual int isAppropriate(TLEventSubRegion* );
```

Returns TRUE if subregion should have a view on me. Default behavior is to check subregion name against mySubRegionName Overrides TLEvent::isAppropriate()

- maxVal

```
virtual double maxVal() const;
```

Get maximum values to be displayed

- maxVal

```
virtual void maxVal(double );
```

Set maximum values to be displayed

- minVal

```
virtual void minVal(double );
```

Set minimum values to be displayed

- minVal

```
virtual double minVal() const;
```

Get minimum values to be displayed

- name

```
virtual const HString& name() const;
```

Get name - base class behavior is to return a reference to an empty string. PIRpTlValEvents do not display text on views

- operator =

```
PIRpTlValEvent& operator =(const PIRpTlValEvent& );
```

assignment operator disabled - nothing is assigned

- subRegName

```
virtual const HString& subRegName() const;
```

Set name of subregion I should be displayed on

- subRegName

```
virtual void subRegName(const char* );
```

Get name of subregion I should be displayed on

- thldVal

```
virtual double thldVal() const;
```

Get threshold values to be displayed

- thldVal

```
virtual void thldVal(double );
```

Set threshold values to be displayed

- times

```
virtual int times(HEpochTime& st, HEpochTime& en);
```

Return start and end times of total interval over which I am displaying data.

- type

```
virtual const HString& type() const;
```

Set type for color mapping, etc.

- type

```
virtual void type(const char* );
```

Get type for color mapping, etc.

- updateVal

```
virtual int updateVal(const HEpochInterval& intvl);
```

new data over specified time interval. tell my view to update itself over the interval.

- val

```
virtual int val(const HEpochInterval& reqIntvl, HObjCollection& col);
```

Fill the collection with the data that is to be displayed over the requested interval. If data is put into collection, TRUE is returned. Protocol only - the exact nature of the data depends on derived classes.

6.3.24.3.29 PIRpTIVALView Class

Overview:

This class is the base class for views that display value versus time information graphically in a timeline subregion.

Export Control: Public

Inheritance Relationships:

Attributes:

myOvrThldColor: DRGBColor

My color for displaying over-threshold data values

myVisIntvl: HEpochInterval

The currently visible portion of the total interval over which I am displaying data

myVisRect: HRectangle

The rectangle on the display which my currently visible interval covers

Constructors and Destructor:

```
PlRpTlValView(TLEvent* );
```

constructor. argument is owning TLEvent.

```
PlRpTlValView(const PlRpTlValView& );
```

copy constructor disabled - nothing is copied

```
~PlRpTlValView();
```

destructor

Operations:

- accDrag

```
virtual unsigned int accDrag();
```

return FALSE because default is to not accept drags overrides base class function

- adjPos

```
virtual void adjPos(long , long );
```

adjPos does nothing because default is to not be able to drag a PlRpTlValView. overrides base class functions

- containedBy

```
virtual unsigned int containedBy(HRectangle* ) const;
```

contains check myVisRect rectangle overrides base class functions

- contains

```
virtual unsigned int contains(HRectangle* ) const;
```

contains check myVisRect rectangle overrides base class functions

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlValView );
```

This class is the base class for views that display value versus time information graphically in a timeline subregion.

- erase

```
virtual void erase();
```

erase whole view basically tell subregion to paint over myVisRect overrides base class function

- findEventRegion

```
virtual TLEventSubRegion* findEventRegion();
```

findEventRegion just returns the subregion I am on, since default is to not be able to move a PlRpTlValView overrides base class function

- getPos

```
virtual void getPos();
```

getPos does nothing because default is to not be able to drag a PlRpTlValView. overrides base class functions

- intersects

```
virtual unsigned int intersects(const HRectangle* ) const;
```

intersection, check myVisRect rectangle overrides base class functions

- intersects

```
virtual unsigned int intersects(unsigned long x, unsigned long y) const;
```

intersection, check myVisRect rectangle overrides base class functions

- operator =

```
PlRpTlValView& operator =(const PlRpTlValView& );
```

assignment operator disabled - nothing assigned

- ovrThldColor

```
virtual const DRGBColor& ovrThldColor() const;
```

get my over-threshold color

- ovrThldColor

```
virtual void ovrThldColor(const DRGBColor& col);
```

set my over-threshold color

- pixel

```
virtual unsigned long pixel(double val);
```

determine the pixel location corresponding to the input value by doing a linear interpolation based on myVisRect and the max and min values to be displayed as obtained from the associated PIRpTIValevent

- recalcVisIntvl

```
virtual void recalcVisIntvl(TLTimeScaleAbs* );
```

recalculate my visible interval

- recalcVisRect

```
virtual void recalcVisRect(TLTimeScaleAbs* );
```

recalculate my visible rectangle

- recalc

```
virtual void recalc(TLTimeScaleAbs* );
```

recalculate my visible interval and my current bounding rectangle on display based on my total interval and the given timescale. overrides base class function

- repaintSpecifics

```
virtual int repaintSpecifics(const HObjCollection& );
```

repaint what I am displaying using the data in the HObjCollection. protocol only - derived classes must implement based on kind of data they expect in the HObjCollection and the kind of view they are presenting.

- repaint

```
virtual int repaint();
```

repaint by getting data over myVisIntvl from my associated PIRpTIValevent and then calling repaintSpecifics overrides base class function

- savePos

```
virtual void savePos();
```

savePos does nothing because default is to not be able to drag a PIRpTIValevent. overrides base class functions

- selectView

```
virtual void selectView();
```

for individual select, unselect, do nothing, because default is to not let PIRpTIValevents be selected overrides base class functions

- unselectView

```
virtual void unselectView();
```

- updateVal

```
virtual int updateVal(const HEpochInterval& intvl);
```

new data over specified interval. if this interval overlaps myVisIntvl, erase myself and repaint to show new data

- visInterval

```
virtual const HEpochInterval& visInterval();
```

return my visible interval overrides base class function

6.3.24.3.30 PIRpTIWindow Class

Overview:

PIRpTIWindow are main windows used by the PIRpTlAppl class.

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

```
PIRpTlWindow();
```

Constructor

```
~PIRpTlWindow();
```

Destructor

Operations:

- createDisplay

```
virtual int createDisplay();
```

create the display. here just create the colormap

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpTlWindow );
```

PIRpTIWindow are main windows used by the PIRpTlAppl class.

- `initIcon`

```
virtual void initIcon();
```

Create a planning timeline icon. Overrides DWindow::initIcon

6.3.24.3.31 PIRpTlXtl Class

Overview:

Instances of PIRpTlXtl are planning timeline displays. A PIRpTlXtl is the main application class for the timeline. It is responsible for setting up communication with the name server process to communicate with a system resource model(SRM) which contains a resource pool from which it gets the resource states to display.

Export Control: Public

Inheritance Relationships:

Attributes:

```
myBitFile: HString
```

bitmap file that contains path and file name for stipple bitmap

```
myDefsConfig: PlRpTlDefsConfig*
```

my configuration file for various default values.

```
myDispIntvl: HEpochInterval
```

my displayed interval

```
myParent: PlRpTlAppl*
```

parent PIRpTlAppl

```
myPlRpTlStatusBar: PlRpTlStatusBar*
```

my PIRpTlXtl Status bar

```
myPlRpTlZmBar: PlRpTlZmBar*
```

my PIRpTlXtl zoom bar

```
myStplConfig: PlRpTlStplConfig*
```

my configuration file for stipple files.

```
myTypesConfig: PlRpTlTypesConfig*
```

my configuration file for types.

Constructors and Destructor:

```
PlRpTlXtl(PlRpTlAppl* );
```

constructor - parent Appl as parameter

```
PlRpTlXtl(const PlRpTlXtl& orig);
```

copy constructor disabled - nothing is copied

```
~PlRpTlXtl();
```

destructor

Operations:

- actTypNm

```
RWCString actTypNm(int id);
```

fetch the reservation activity type name string by id

- changeViewedRs

```
virtual int changeViewedRs(const HObjCollection& rsToDelete);
```

change my viewed resources to match new set provided by PlRpTlAppl. argument is list of currently-viewed resources that I don't want to see anymore.

- clearInternals

```
virtual void clearInternals();
```

prepare for a configuration change, ask all entities to clear so we can totally reconfigure asks my own entities, then calls base class behavior overrides void clearInternals()

- configBasicDefaults

```
virtual int configBasicDefaults();
```

set up basic defaults like region colors and time interval

- configStpl

```
virtual int configStpl();
```

set up possible event types and their associated stipple file

- configSubRegions

```
virtual int configSubRegions();
```

set up a subregion and a label for each resource that is to be displayed on the timeline

- configTypes

```
virtual int configTypes();
```

set up possible event types and their associated colors

- createDefsConfig

```
virtual PIRpTlDefsConfig* createDefsConfig();
```

create my config file for loading/saving various defaults

- createDisplay

```
virtual int createDisplay();
```

create the display. overrides Xtl :: createDisplay to add createStatusBar

- createMainReg

```
virtual int createMainReg();
```

Create main region. overrides Xtl :: createMainReg to return PIRpTlMainRegion type instead of TLMainRegion type

- createNewPtlEvents

```
virtual void createNewPtlEvents(RResource* rs, unsigned int key,  
const char* nm, const HEpochInterval& spanOfUpdate);
```

create principal timeline events for resource

- createNewPtlEvent

```
virtual PIRpTlSimpleEvent* createNewPtlEvent();
```

allocates a PIRpTlSimpleEvent from free store using "this"

- createNewStplEvents

```
virtual void createNewStplEvents(RResource* rs, unsigned int  
key, const char* nm, const HEpochInterval& spanOfUpdate);
```

create stipple events for resource

- createNewStplEvent

```
virtual PIRpTlStplEvent* createNewStplEvent(RResource* rs);
```

allocates a PIRpTlStplEvent from free store using "this"

- createPanel

```
virtual int createPanel();
```

create the PIRpTlXtl main panel and menus. this adds facilities to permit interactive changes to configuration, colors, and resources. overrides Xtl::createPanel()

- createScalableFonts

```
virtual DFontXScalable* createScalableFonts();
```

Create default scalable fonts overrides Xtl::createScalableFonts()

- createStatusBar

```
virtual int createStatusBar();
```

create an PIRpTIXtl status bar for display messages

- createStplConfig

```
virtual PIRpTlStplConfig* createStplConfig();
```

create my config file for loading/saving types and stipple pattern mappings

- createSubRegion

```
virtual int createSubRegion(const HString& resNm, DFontX* font);
```

create up a subregion and a label for the given name to be displayed on the timeline

- createTypesConfig

```
virtual PIRpTlTypesConfig* createTypesConfig();
```

create my config file for loading/saving types and type-to-color mappings

- createZoomBar

```
virtual int createZoomBar();
```

create an PIRpTIXtl zoom bar for horizontal scrolling overrides Xtl::createZoomBar()

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PIRpTlXtl );
```

Instances of PIRpTIXtl are planning timeline displays. A PIRpTIXtl is the main application class for the timeline. It is responsible for setting up communication with the name server process to communicate with a system resource model(SRM) which contains a resource pool from which it gets the resource states to display.

- defsConfig

```
virtual PIRpTlDefsConfig* defsConfig();
```

get my default configuration information

- determineConfig

```
virtual int determineConfig();
```

determine initial configuration setup overrides Xtl::determineConfig

- dispIntvl

```
virtual const HEpochInterval& dispIntvl() const;
```

get my display interval

- dispIntvl

```
virtual void dispIntvl(const HEpochInterval& intvl);
```

set/get my display interval

- doAttachments

```
virtual int doAttachments();
```

set up the proper attachments. overridden to operate on the PIRpTlXtl zoombar overrides Xtl::doAttachments()

- doManages

```
virtual int doManages();
```

manage the timeline area. overridden to manage the PIRpTlXtl zoom bar overrides Xtl::doManages()

- getFont

```
virtual DFontX* getFont(const char* family, const char* style,  
int size);
```

Obtain a label font of the desired family, style, and size

- getPIRpTlPanel

```
virtual PlRpTlPanel* getPlRpTlPanel();
```

get the PIRpTlPanel - I do not pass on ownership

- initialize

```
virtual int initialize();
```

initialize the timeline display overrides Xtl::initialize

- loadDefaults

```
virtual int loadDefaults();
```

load my default values by creating the defs configuration file and asking it to configure

- operator =

```
PlRpTlXtl& operator =(const PlRpTlXtl& orig);
```

assignment operator disabled - nothing is assigned

- ptlStatusBar

```
virtual PlRpTlStatusBar* ptlStatusBar();
```

get the PIRpTlXtl zoom bar area

- ptlZmBar

```
virtual PIRpTlZmBar* ptlZmBar();
```

get the PIRpTlXtl zoom bar area

- reconfigure

```
virtual int reconfigure();
```

reconfigure myself for a new config file

- removeImpactedPtlEvents

```
virtual void removeImpactedPtlEvents(unsigned int key, const char* rsrcNm, const HEpochInterval& spanOfUpdate);
```

remove principal timeline events from updated resource

- removeImpactedStplEvents

```
virtual void removeImpactedStplEvents(unsigned int key, const char* rsrcNm, const HEpochInterval& spanOfUpdate);
```

remove stipple events from updated resource

- resize

```
virtual int resize(int newWidth, int newHeight);
```

override resize to take of PIRpTlXtl zoom bar

- saveConfig

```
virtual int saveConfig(const char* fileNm);
```

save my configuration to the given file

- setDefBoundsAndSizes

```
virtual int setDefBoundsAndSizes();
```

Set the default width and height of the principal timeline regions overrides Xtl::setDefBoundsAndSizes()

- snapToConfig

```
virtual int snapToConfig();
```

override snapToConfig to take care of PIRpTlXtl zoom bar

- snapToInterval

```
virtual void snapToInterval(const HEpochInterval& );
```

override snapToInterval to take care of PIRpTlXtl zoom bar

- stateUpdate

```
virtual void stateUpdate(RUpdState* anUpd);
```

called when my noid gets notification of a state update. I get the list of updates.

- switchTotalIntvl

```
virtual int switchTotalIntvl(const HEpochInterval& intvl);
```

switch my overall interval to the given interval. I will display the same amount of time I am showing now (if I can), but it will start at the beginning of the new total interval.

- updStates

```
virtual void updStates(RResource* rs, const char* rsrcNm,  
unsigned int key, RUpdState* thisUpd);
```

update a resource that is being displayed as a set of states

6.3.24.3.32 PIRpTIZmBar Class

Overview:

Instances of PIRpTIZmBar are zoom bars which know how to pop up a plan interval window.

Export Control: Public

Inheritance Relationships:

Attributes:

myDurDrag: int

internal variable, TRUE during drag FALSE otherwise

myMenu: DOptionMenu*

my option menu

myOptionBtns: HObjList

my list of push buttons on the option menu kept so I can destroy them during reconfiguration

myScrollbar: PIRpTlZmBarCbScrollbar*

my scrollbar

mySliderPup: PIRpTlSliderPup*

my slider popup

myXtl: Xtl*

my owning xtl

Constructors and Destructor:

```
PlRpTlZmBar(Xtl* );
```

constructor. given a pointer to my owning xtl

```
PlRpTlZmBar(const PlRpTlZmBar& orig);
```

copy - disabled

```
~PlRpTlZmBar();
```

destructor

Operations:

- adjustSlider

```
virtual int adjustSlider(const HEpochInterval& dispIntvl, const  
HEpochInterval& totIntvl);
```

adjust slider position and length to match relationship between given displayed interval and given total interval

- adjustToNewIntvl

```
virtual int adjustToNewIntvl(int durSecs);
```

adjust scrollbar to reflect new interval duration and snap ptl to new displayed interval. I make sure the new interval stays within my total interval.

- clear

```
virtual void clear();
```

clear.

- computeSliderIntvl

```
virtual void computeSliderIntvl(HEpochInterval& intvl);
```

compute display interval based on current slider position. return it in intvl

- createOptionsMenu

```
virtual int createOptionsMenu();
```

create option menu

- createScrollbar

```
virtual int createScrollbar();
```

create scrollbar

- createSliderPup

```
virtual int createSliderPup();
```

create slider popup, and hide it for now

- DECLARE_LOCAL_CLASS

```
int DECLARE_LOCAL_CLASS(PlRpTlZmBar );
```

Instances of PlRpTlZmBar are zoom bars which know how to pop up a plan interval window.

- fillOptionsMenu

```
virtual int fillOptionsMenu();
```

fill the option menu from the defaults file

- getDefsConfig

```
virtual PlRpTlDefsConfig* getDefsConfig();
```

return the default configuration class

- init

```
virtual int init();
```

initialize. create the form and its child scrollbar and option menu

- operator =

```
PlRpTlZmBar& operator =(const PlRpTlZmBar& orig);
```

assignment - disabled

- positionWidgets

```
virtual int positionWidgets();
```

position widgets

- reconfigure

```
virtual int reconfigure();
```

reconfigure. reinitialize the option menu buttons

- scrollDragCb

```
virtual void scrollDragCb(PlRpTlZmBarCbScrollbar* );
```

callback when scrollbar is dragged

- scrollValChgCb

```
virtual void scrollValChgCb(PlRpTlZmBarCbScrollbar* );
```

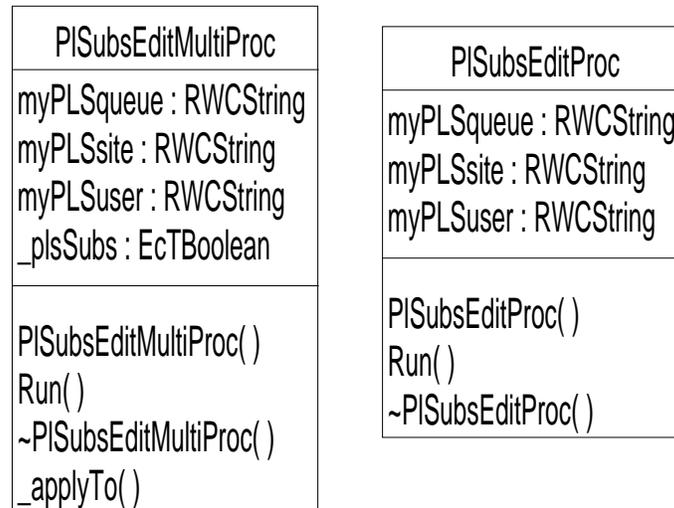
callback when scrollbar value is changed

- showSelCb

```
virtual void showSelCb(PlRpTlZmBarCbPushBtn* btn);  
callback when new show percentage is selected
```

6.3.25 Planning_Subscription_Editor Class Category

6.3.25.1 Overview



6.3.25.2 Planning_Subscription_Editor

6.3.25.3 PISubsEditMultiProc

Overview:

Export Control:

Inheritance Relationships:

Inherits from

Attributes:

`myPLSqueue: RWCString`

`myPLSsite: RWCString`

`myPLSuser: RWCString`

`_plsSubs: EcTBoolean`

Constructors and Destructor:

`PLSubsEditMultiProc();`

constructor

`virtual ~PLSubsEditMultiProc();`

destructor

Operations:

- Run

`EcUtStatus Run();`

main method to enter a subscription

- _applyTo

`EcUtStatus _applyTo(const RWCString& , const GIClient* thisClient);`

6.3.25.3.1 PISubsEditProc Class

Overview:

Export Control: Public

Inheritance Relationships:

Inherits from EcPfGenProcess

Attributes:

```
myPLSqueue: RWCString
```

```
myPLSsite: RWCString
```

```
myPLSuser: RWCString
```

Constructors and Destructor:

```
PLSubsEditProc();
```

constructor

```
virtual ~PLSubsEditProc();
```

destructor

Operations:

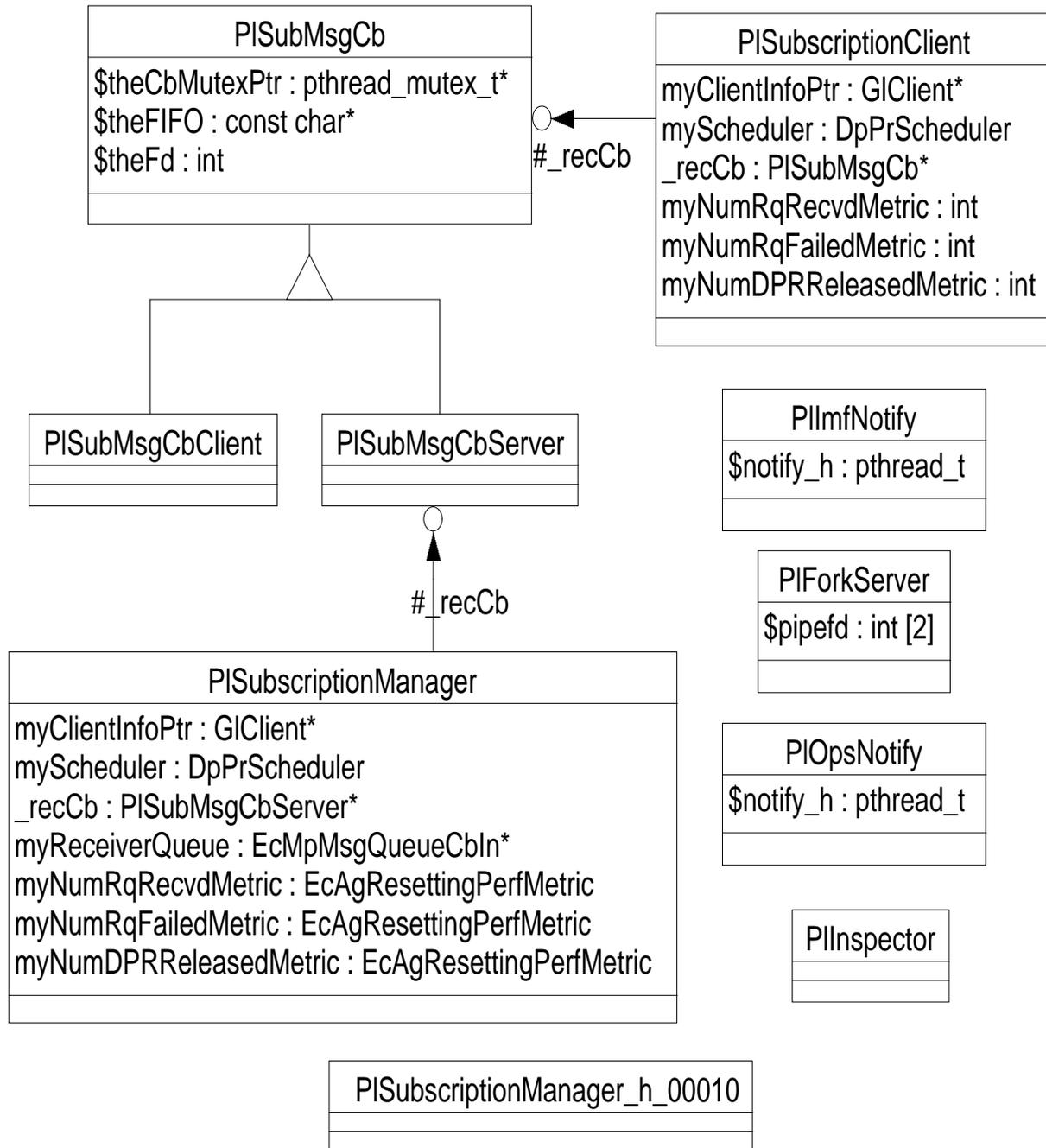
- Run

```
EcUtStatus Run();
```

main method to enter a subscription

6.3.26 Planning_Subscription_Manager Class Category

6.3.26.1 Overview



6.3.26.2 Planning_Subscription_Manager Classes

6.3.26.3 PIForkServer Class

Overview:

encapsulate the logic that forks the subscription manager into two procs. the parent proc remains the dce server and performs the listen, its callback handler writes notification urs to the child via the internal pipe the child awaits piped i/o and calls the (original) fun&games callback

Export Control: Public

Inheritance Relationships:

Attributes:

```
pipefd: int [2]
```

Constructors and Destructor:

```
PIForkServer( );
```

encapsulate the logic that forks the subscription manager into two procs. the parent proc remains the dce server and performs the listen, its callback handler writes notification urs to the child via the internal pipe the child awaits piped i/o and calls the (original) fun&games callback

Operations:

- childAction

```
static int childAction(int fd, EcTInt argc, EcTChar** argv,  
EcUtStatus& recStatus);
```

- doForkExec

```
static int doForkExec(const char* newproc, EcTInt argc, EcTChar**  
argv, EcUtStatus& recStatus);
```

- doFork

```
static int doFork(EcTInt argc, EcTChar** argv, EcUtStatus&
recStatus);
```

- parentAction

```
static int parentAction(int fd, EcTInt argc, EcTChar** argv,
EcUtStatus& recStatus);
```

6.3.26.3.1 PllmfNotify Class

Overview:

encapsulate the imf polling scheme, imf insert notifications are ur strings placed into getenv("DataServer")/.notify directory entries (empty files)

Export Control: Public

Inheritance Relationships:

Attributes:

notify_h: pthread_t

encapsulate the imf polling scheme, imf insert notifications are ur strings placed into getenv("DataServer")/.notify directory entries (empty files)

Constructors and Destructor:

Operations:

- fetchURList

```
static void fetchURList(const RWCString& unreleased, RWOrdered&
urlist);
```

- handleNotifications

```
static void handleNotifications(PlSubMsgCb* cb);
```

process multiple entries in .notify direccory:

- notify

```
static void notify(pthread_addr_t p);
```

infinite loop

- poll

```
static pthread_t poll(pthread_addr_t p);
```

creates the thread that exexutes notify

- retryFailedNotifications

```
static void retryFailedNotifications(P1SubMsgCb* cb);
```

- Try

```
static void Try(P1SubMsgCb* cb, const RWCString& notifydir, const  
RWCString& unreleasedir, const RWCString& notice);
```

6.3.26.3.2 P1Inspector Class

Overview:

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

Operations:

- examineParmList

```
static void examineParmList(const RWTPtrOrderedVector&  
attributes, const G1ParameterList& parmList);
```

- getStringParm

```
static RWCString getStringParm(const RWCString& ur, const
RWCString& p);
```

6.3.26.3.3 PIOpsNotify Class

Overview:

encapsulate the imf polling scheme, imf insert notifications are ur strings placed into getenv("DataServer")/.notify directory entries (empty files)

Export Control: Public

Inheritance Relationships:

Attributes:

notify_h: pthread_t

encapsulate the imf polling scheme, imf insert notifications are ur strings placed into getenv("DataServer")/.notify directory entries (empty files)

Constructors and Destructor:

Operations:

- handleNotifications

```
static void handleNotifications(P1SubMsgCbClient* cb);
```

process multiple entries in .notify directory:

- notify

```
static void notify(pthread_addr_t p);
```

infinite loop

- poll

```
static pthread_t poll(pthread_addr_t p);
```

creates the thread that exexutes notify

6.3.26.3.4 PISubMsgCbClient Class

Overview:

This class provides the main command logic to the subscription manager. The class derives from the CSS message passing callback and is invoked on reception of a subscription notification from DSS

Export Control: Public

Inheritance Relationships:

Inherits from PISubMsgCb

Attributes:

Constructors and Destructor:

```
PISubMsgCbClient();
```

Constructor

```
virtual ~PISubMsgCbClient();
```

Destructor

Operations:

- HandleCbMsg

```
virtual EcUtStatus HandleCbMsg(const EcTPtr msgP, const EcTUShortInt msgRWClassId, const EcTULongInt msgLength, const RWCString& msgId, const RWCString& replyMsgId, const RWCString& senderLogName, const RWCString& senderSiteName, const RWCString& recverLogName);
```

Callback function, overrides CSS message passing callback virtual method. This method contains the main command logic that manages the processing that is invoked on reception of the subscription notification

6.3.26.3.5 PISubMsgCbServer Class

Overview:

This class provides the main command logic to the subscription manager. The class derives from the CSS message passing callback and is invoked on reception of a subscription notification from DSS

Export Control: Public

Inheritance Relationships:

Inherits from `PISubMsgCb`

Attributes:

Constructors and Destructor:

```
PISubMsgCbServer();
```

Constructor

```
virtual ~PISubMsgCbServer();
```

Destructor

Operations:

- `HandleCbMsg`

```
virtual EcUtStatus HandleCbMsg(const EcTPtr msgP, const  
EcTUShortInt msgRWClassId, const EcTULongInt msgLength, const  
RWCString& msgId, const RWCString& replyMsgId, const RWCString&  
senderLogName, const RWCString& senderSiteName, const RWCString&  
recverLogName);
```

Callback function, overrides CSS message passing callback virtual method. This method contains the main command logic that manages the processing that is invoked on reception of the subscription notification

- `_examineHeader`

```
EcUtStatus _examineHeader(const RWCString& message, RWCString&  
urString) const;
```

- `_handleCbFull`

```
virtual EcUtStatus _handleCbFull(const EcTPtr msgP, const  
EcTUShortInt msgRWClassId, const EcTULongInt msgLength, const  
RWCString& msgId, const RWCString& replyMsgId, const RWCString&  
senderLogName, const RWCString& senderSiteName, const RWCString&  
recverLogName);
```

Receiver Logical Name

- `_handleCbStdOut`

```
virtual EcUtStatus _handleCbStdOut(const EcTPtr msgP, const  
EcTUShortInt msgRWClassId, const EcTULongInt msgLength, const
```

```
RWCString& msgId, const RWCString& replyMsgId, const RWCString&
senderLogName, const RWCString& senderSiteName, const RWCString&
recverLogName);
```

Receiver Logical Name

- `_handleCbUsrTmpDirent`

```
virtual EcUtStatus _handleCbUsrTmpDirent(const EcTPtr msgP,
const EcTUShortInt msgRWClassId, const EcTULongInt msgLength,
const RWCString& msgId, const RWCString& replyMsgId, const
RWCString& senderLogName, const RWCString& senderSiteName, const
RWCString& recverLogName);
```

Receiver Logical Name

6.3.26.3.6 PISubMsgCb Class

Overview:

This class provides the main command logic to the subscription manager. The class derives from the CSS message passing callback and is invoked on reception of a subscription notification from DSS

Export Control: Public

Inheritance Relationships:

Attributes:

```
theCbMutexPtr: pthread_mutex_t*
```

```
theFd: int
```

```
theFIFO: const char*
```

Receiver Logical Name

Constructors and Destructor:

```
PISubMsgCb();
```

Constructor

```
virtual ~PISubMsgCb();
```

Destructor

Operations:

- HandleCbMsg

```
virtual EcUtStatus HandleCbMsg(const EcTPtr msgP, const  
EcTUShortInt msgRWClassId, const EcTULongInt msgLength, const  
RWCString& msgId, const RWCString& replyMsgId, const RWCString&  
senderLogName, const RWCString& senderSiteName, const RWCString&  
recverLogName);
```

Callback function, overrides CSS message passing callback virtual method. This method contains the main command logic that manages the processing that is invoked on reception of the subscription notification

- lock

```
static void lock();
```

- Throw

```
static void Throw(const RWCString& s);
```

- Throw

```
static void Throw(const EcUtStatus& s);
```

- unlock

```
static void unlock();
```

6.3.26.3.7 PISubscriptionClient Class

Overview:

This class is the main application class for the subscription manager. It derives from the Process Framework as a fully managed process.

Export Control: Public

Inheritance Relationships:

Inherits from EcPfGenProcess

Attributes:

```
myClientInfoPtr: GlClient*
```

Client information used for Science Data Server interaction

myNumDPRReleasedMetric: int

Metric reporting the number of DPRs released

myNumRqFailedMetric: int

Metric reporting the number of subscriptions that don't match a granule for which we are waiting on

myNumRqRecvdMetric: int

Metric recording the number of subscription notifications received

myScheduler: DpPrScheduler

DpPrScheduler defines the interface to the DPS job scheduler The default scheduler is created as an attribute of the subscription manager. It is initialized in the Start() method so that the Release and Cancel methods can be used in the subscription manager callback without reinitializing the scheduler each time a notification is received

_recCb: P1SubMsgCb*

To use the callback from any thread (specifically, the signal catcher thread), need to keep pointer to it -- dh

Constructors and Destructor:

```
P1SubscriptionClient(EcTInt argc, EcTChar** argv, EcUtStatus* );
```

Constructor

```
~P1SubscriptionClient();
```

Destructor

Operations:

- CancelDprJob

```
EcUtStatus CancelDprJob(P1Dpr& dpr);
```

Interact with the scheduler to Cancel a DPR

- GetClientInfoPtr

```
const G1Client* GetClientInfoPtr();
```

Return client info for Science Data Server interaction

- handleNotice

```
void handleNotice(const RWCString& );
```

- `init`
virtual EcUtStatus init(int fd);
init method, contains main application initialization steps -- dh
- `LogDPRReleased`
EcTVoid LogDPRReleased(const RWCString&);
Log that a DPR has been released from the receipt of a subscription notice
- `LogRqFailed`
EcTVoid LogRqFailed(const RWCString&);
Log that a subscription notice didn't match something we're waiting for
- `LogRqRecvd`
EcTVoid LogRqRecvd(const RWCString&);
Log that a subscription notice has been received
- `LogShutdown`
EcTVoid LogShutdown();
Log shutdown event
- `LogStartup`
EcTVoid LogStartup();
Log startup event
- `recCb`
PlSubMsgCb* recCb();
To use the callback from any thread (specifically, the signal catcher thread), need to return pointer to it -- dh
- `ReleaseDprJob`
EcUtStatus ReleaseDprJob(PlDpr& dpr);
Interact with the scheduler to Release a DPR
- `Start`
virtual EcUtStatus Start();
Startup method delegates to PfStart -- dh

6.3.26.3.8 PISubscriptionManager_h_00010 Class

Overview:

Export Control: Public

Inheritance Relationships:

Attributes:

Constructors and Destructor:

Operations:

6.3.26.3.9 PISubscriptionManager Class

Overview:

This class is the main application class for the subscription manager. It derives from the Process Framework as a fully managed process.

Export Control: Public

Inheritance Relationships:

Attributes:

myClientInfoPtr: GIClient*

Client information used for Science Data Server interaction

myNumDPRReleasedMetric: EcAgResettingPerfMetric

Metric reporting the number of DPRs released

myNumRqFailedMetric: EcAgResettingPerfMetric

Metric reporting the number of subscriptions that don't match a granule for which we are waiting on

myNumRqRecvdMetric: EcAgResettingPerfMetric

Metric recording the number of subscription notifications received

myReceiverQueue: EcMpMsgQueueCbIn*

Pointer to the receiver queue object registered with CSS for callback on receipt of a subscription notification

myScheduler: DpPrScheduler

DpPrScheduler defines the interface to the DPS job scheduler The default scheduler is created as an attribute of the subscription manager. It is initialized in the Start() method so that the Release and Cancel methods can be used in the subscription manager callback without reinitializing the scheduler each time a notification is received

_recCb: PlSubMsgCbServer*

To use the callback from any thread (specifically, the signal catcher thread), need to keep pointer to it -- dh

Constructors and Destructor:

```
PlSubscriptionManager(EcTInt argc, EcTChar** argv, EcUtStatus*  
);
```

Constructor

```
~PlSubscriptionManager();
```

Destructor

Operations:

- CancelDprJob

```
EcUtStatus CancelDprJob(PlDpr& dpr);
```

Interact with the scheduler to Cancel a DPR

- GetClientInfoPtr

```
const GlClient* GetClientInfoPtr();
```

Return client info for Science Data Server interaction

- IncrementMetric

```
EcTVoid IncrementMetric(EcAgPerfMetric* );
```

Increment one of the performance metrics

- init

```
virtual EcUtStatus init();
```

init method, contains main application initialization steps -- dh

- LogDPRReleased

```
EcTVoid LogDPRReleased(const RWCString& );
```

Log that a DPR has been released from the receipt of a subscription notice

- LogRqFailed

```
EcTVoid LogRqFailed(const RWCString& );
```

Log that a subscription notice didn't match something we're waiting for

- LogRqRecvd

```
EcTVoid LogRqRecvd(const RWCString& );
```

Log that a subscription notice has been received

- LogShutdown

```
EcTVoid LogShutdown();
```

Log shutdown event

- LogStartup

```
EcTVoid LogStartup();
```

Log startup event

- PfGetShutdownSec

```
virtual EcTInt PfGetShutdownSec(EcTAgMgmtLevel shutdownLevel);
```

Concrete implementation of the process framework virtual method to provide the number of seconds required to shutdown back to the MSS agents

- PfShutdown

```
virtual EcUtStatus PfShutdown(EcTAgMgmtLevel shutdownLevel,  
EcTInt shutdownReason, EcTInt gracefulFlag);
```

Concrete implementation of the process framework virtual method to respond to a request to shutdown

- recCb

```
PlSubMsgCbServer* recCb();
```

To use the callback from any thread (specifically, the signal catcher thread), need to return pointer to it -- dh

- RegisterMetrics

```
EcUtStatus RegisterMetrics();
```

Register the metrics that will be reported to MSS

- ReleaseDprJob

```
EcUtStatus ReleaseDprJob(P1Dpr& dpr);
```

Interact with the scheduler to Release a DPR

- SetupSecurity

```
EcUtStatus SetupSecurity();
```

Setup the security

- Start

```
virtual EcUtStatus Start();
```

Startup method delegates to PfStart -- dh